

---

# Herramienta de apoyo para la Generación de Resúmenes Fáciles de Leer para personas con dificultades de comprensión

---

Francisco Javier Vabe Bakale

Karen Margot Cano León



Facultad de Informática  
Universidad Complutense de Madrid

Departamento de Ingeniería del Software e Inteligencia Artificial  
Curso 2016/2017

Director:  
Alberto Díaz Esteban

# *Autorización de difusión*

*Francisco Javier Vabe Bakale y Karen Margot Cano León, autorizamos a la universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria como el código, la documentación y/o aplicación desarrollada.*

---

*Francisco Javier Vabe Bakale*

---

*Karen Margot Cano León*

# *Agradecimiento*

Queremos agradecer a todas aquellas personas que nos han ayudado a llevar a cabo este proyecto, mención especial a nuestro tutor: Alberto Díaz Esteban, director del proyecto por guiarnos y apoyarnos, por sus consejos y ánimos. A Antonio F. G. Sevilla, creador de Grafeno, por ayudarnos a comprender su funcionamiento y prepararnos los servidores web.

A los profesores por sus enseñanzas y entrega en su labor de compartir sus conocimientos, pues sin ellos no hubiese sido posible realizar este proyecto fin de carrera.

Agradecer a nuestras familias por habernos facilitado la posibilidad de finalizar nuestra formación académica y porque sin su apoyo no hubiese sido posible recorrer este camino.

Por último, agradecer a mis compañeros, por haber formado parte de este recorrido y por su apoyo siempre que lo hemos necesitado.

# Resumen

Hoy en día, la gran cantidad de información disponible, sobre todo en Internet, hace que sea prácticamente imposible leerla y entenderla por completo. Vivimos en una sociedad de constantes cambios tecnológicos, las tecnologías de la información evolucionan y cambian nuestra forma de vida en muchas áreas de conocimiento. Estos cambios hacen necesarios desarrollos de nuevas aplicaciones relacionadas con el Procesamiento del Lenguaje Humano, cuyos objetivos sean facilitar la comprensión de las personas, tengan o no algún tipo de discapacidad de comprensión.

Este proyecto fin de grado tiene como finalidad unir varias tecnologías con el fin de poder cumplir un claro objetivo: desarrollar una aplicación que permita a las personas con algún tipo de dificultad de comprensión tener acceso a la información de una forma más sencilla y fácil de comprender.

Este proyecto consta de dos partes bien diferenciadas e independientes. En la primera parte, se desarrolla la funcionalidad para la generación del resumen de un texto en español. Para esta funcionalidad se hace uso de la librería [Grafeno](https://github.com/agarsev/grafeno)<sup>1</sup>, la cual extrae grafos semánticos a partir de contenidos textuales y nos proporciona los recursos necesarios para que el procesamiento de la información sea más sencillo.

En la segunda parte se desarrolla la funcionalidad de simplificación de textos. Esta funcionalidad transforma un texto en otro equivalente, más fácil de entender por una audiencia determinada. Las oraciones complejas se transforman en oraciones más simples y se reemplazan las palabras difíciles por otras más sencillas. Para esto, se hace uso de distintas herramientas de código abierto ya existentes.

Las dos funcionalidades descritas anteriormente pueden ser usadas de forma conjunta o independiente. Esta característica nos facilita la tarea de usar la una o la otra. Nuestra herramienta cuenta con una interfaz sencilla e intuitiva que facilita su acceso a personas con pocos conocimientos en informática.

---

<sup>1</sup> <https://github.com/agarsev/grafeno>

# *Abstract*

Currently we have a large amount of information available especially on the Internet, that means that it is almost impossible to read and process it according to our interests. Nowadays, we live in a period of constant technological changes and the improvements on information technology have evolved and changed our everyday lives and many areas of knowledge. For that reason, new Natural Language Processing applications are needed in order to ease people's life, whether they have a disability or not.

This final project aims to join several technologies in order to achieve a clear goal. The development of an application that allows people with some type of comprehension difficulty to access information in a simpler and easier way to understand.

This project consists of two clearly differentiated and independent parts. In the first part, the functionality for the generation of a summary of a text in Spanish is developed. We have used Grafeno library, which provides us with the resources to process the information in a simpler way.

In the second part, the functionality of the simplification of a text is developed. This consists of transforming a text into an equivalent that is easier to read by a particular group of people. Complex sentences are transformed into simpler sentences and difficult words are replaced with easier words using different open code tools.

Both functionalities can be used jointly or independently, which increases flexibility in carrying out one task or another. Our application has an easy and intuitive interface even for people with little computer knowledge.

## *Palabras claves*

- Accesibilidad
- Dependencias
- Grafeno
- Procesamiento de texto
- Resumen
- Simplificación

# *Keywords*

- Accessibility
- Dependencies
- Grafeno
- Word processing
- Summary
- Simplification

# Índice General

<b>Capítulo 1: Introducción</b>	<b>11</b>
1.1 Motivación	12
1.2 Objetivos	13
1.3 Estructura del Documento	14
<b>Capítulo 2: Estado del Arte</b>	<b>15</b>
2.1 Introducción	15
2.2 Procesamiento de Lenguaje Natural (PLN)	16
2.3 Accesibilidad a la información	17
2.3.1 La Lectura Fácil	18
2.3.2 Accesibilidad de contenidos Web	21
2.3.3 Simplificación de textos	22
2.3.4 Proyecto Simplext y Able-to-Include	24
2.4 Herramientas de apoyo	29
2.4.1 Grafeo	29
2.4.2 FreeLing	32
2.4.3 WordNet	35
<b>Capítulo 3: Generación del Proyecto</b>	<b>36</b>
3.1 Fase Inicial	36
3.1.1 Objetivos	36
3.1.2 Desarrollo	37
3.1.3 Resultados y discusión crítica	38
3.2 Fase Final	38
<b>Capítulo 4: Generación de Resúmenes</b>	<b>39</b>
4.1 Objetivos	39
4.2 Desarrollo	39
4.3 Resultados y discusión crítica	42



<b>Capítulo 5: Simplificación</b>	<b>44</b>
5.1 Desarrollo	44
5.2 Resultados y discusión crítica	48
<b>Capítulo 6: Evaluación de la Aplicación</b>	<b>50</b>
6.1 Comparación de HAGRes con Simplext	50
6.2. Resultados de la evaluación usando HAGRes y Simplext	52
<b>Capítulo 7: Conclusiones y trabajos futuros</b>	<b>53</b>
7.1 Conclusiones	53
7.2 Trabajos futuros	54
<b>Capítulo 8: Bibliografía y Referencias</b>	<b>55</b>
<b>Apéndice 1</b>	<b>56</b>
<b>Manual de usuario</b>	<b>56</b>
Ap 1.1 Interfaz gráfica	56
Ap 1.2 Ejecución por consola de Eclipse	63
Ap 1.3 Ejecución por consola de comandos	65



# Capítulo 1

## Introducción

La lectura es uno de los procesos más complejos a los que se enfrenta el ser humano. Aprender a leer es una tarea delicada, determinante y exclusiva del ser humano y para desarrollar esa capacidad deberá adquirir una serie de habilidades de alto nivel cognitivo cuya complejidad se refleja tanto a la hora de producir lenguaje como de interpretarlo.

Cuando la acción comunicativa se lleva a cabo entre una persona y una máquina o dispositivos, estos deben ser capaces de manejar el lenguaje humano, por lo que se hace necesario el manejo de técnicas de Análisis Lingüístico y Computacional. Dentro de esta disciplina, el campo que se encarga de la formulación e investigación de mecanismos computacionales que posibiliten dicha comunicación por medio de voz o texto, se denomina Procesamiento de Lenguaje Natural (PLN) [\[1\]](#).

Este proyecto de fin de grado se ha desarrollado para facilitar la accesibilidad a los textos a cualquier persona que pueda tener algún tipo de dificultad, así como servir de apoyo al proceso de elaboración y simplificación de textos escritos en español, a cualquier agente que intervenga en el proceso de comunicación.

En la actualidad podemos ver que los proyectos existentes de resumen y simplificación de textos son escritos y desarrollados en inglés, por esta razón supone una oportunidad, la posibilidad de realizar un proyecto relacionado con la generación de resúmenes y simplificación de textos fáciles de leer, pero en esta ocasión dirigido a personas hispanohablantes.

## 1.1 Motivación

La motivación viene derivada de la necesidad de facilitar la comprensión de textos a aquellas personas que puedan tener algún tipo de dificultad a la hora llevar a cabo esta tarea.

Hoy en día, una de las dificultades que más incidencia puede estar teniendo en el desarrollo de las personas es la relativa a la comprensión de textos escritos. Estamos inmersos en una sociedad, denominada de la información, en la que se producen textos de toda índole y condición, por tanto, para desenvolvernó en la misma, es necesario en la toma de decisiones, una adecuada comprensión de la información que nos llega a través de muy diversas fuentes.

Debido a lo anteriormente mencionado, la sociedad demanda cada vez más la integración completa y efectiva de las personas que presentan algún tipo de dificultad, ya sea transitoria o permanente, para llevar a cabo las distintas tareas cotidianas que conlleva la interpretación de la información que nos llega de forma escrita.

Siendo conscientes de esta necesidad, en la Facultad de Informática de la UCM, a lo largo de los años, se han realizado diferentes proyectos de gran utilidad para la sociedad, todos ellos desarrollados en distintos lenguajes, formatos y estilos. Entre estos proyectos podemos mencionar: *Generación de Lenguaje Natural a partir de Grafos Semánticos* [\[1\]](#), *API de servicios web orientados a la accesibilidad* [\[3\]](#), *Grafeno Semantic Graph Extraction and Operation* [\[4\]](#), entre muchos más. Este último servirá como punto de partida para el desarrollo de nuestro proyecto.

Se pretende dar un valor añadido y una utilidad social a los proyectos anteriormente mencionados, unificándolos y haciéndolos más accesibles mediante una aplicación de escritorio, todo ello dirigido a la población hispanohablante.

Cabe señalar que este proyecto no hubiera sido posible sin el trabajo previo realizado por: Antonio F. G. Sevilla y el grupo de investigación **NIL** (Natural Interaction based on Language), el cual se puede encontrar en un repositorio público de GitHub [\[5\]](#). Este trabajo ha facilitado el desarrollo de este proyecto fin de grado, cuyo origen deriva de la librería **Grafeno**, desarrollada en el marco de un proyecto europeo para obtener y manipular representaciones en forma de grafos semánticos a partir de contenidos textuales.

## 1.2 Objetivos

Este proyecto tiene como objetivo principal desarrollar una Herramienta de Apoyo para la Generación de Resúmenes de Texto Fáciles de Leer en Español dirigida a las personas con dificultades de comprensión, denominada de ahora en adelante (**HAGres**). Esta herramienta contará con dos funcionalidades, las cuales se mencionan a continuación:

- **Generación de resúmenes de texto por extracción:** dado un texto de entrada y haciendo uso de la librería Grafeno, la herramienta generará como salida la extracción de la información más importante del texto.
- **Simplificación de un texto:** dado un texto de entrada, la herramienta construirá una versión simplificada del mismo, sustituyendo las palabras difíciles de comprender por sinónimos más sencillos.

El primer paso ha consistido en la adaptación de Grafeno al español ya que esta librería fue desarrollada para el uso de textos en inglés. Para este proceso Grafeno ha hecho uso de herramientas existentes tales como Freeling, librería configurable y de código abierto que proporciona servicios de análisis lingüísticos para diferentes idiomas.

El segundo paso ha consistido en la generación de resúmenes por extracción haciendo uso de Grafeno. Esta librería se basa en el PLN, cuyo objetivo es transformar el texto objeto de análisis para obtener un resultado lo más modular y flexible posible. Para ello se ha realizado el estudio de los procesos que se llevan a cabo durante la extracción, análisis y transformación de la información.

El tercer paso ha consistido en realizar la simplificación de un texto, transformando las palabras de difícil comprensión por sinónimos u otras palabras que resulten más fácil de entender. Para ello se ha hecho uso de herramientas de código abierto ya existentes.

Una vez realizados los pasos mencionados anteriormente, se obtiene el objetivo principal del proyecto: unificar los recursos para facilitar la accesibilidad de la información a través de resúmenes y simplificación de textos para hispanohablantes. Esta unificación dio como resultado a HAGres, una herramienta que sirve de apoyo para la generación de resúmenes y que cuenta con una interfaz sencilla e intuitiva, que permite el fácil acceso al usuario.

## 1.3 Estructura del Documento

Este documento está dividido en diversos apartados a modo de capítulos que quedan explicados en detalle a continuación.

En el capítulo 2, se explica el Estado del Arte o estado de la cuestión, el cual nos permitirá tener una visión sobre la situación actual de la que se parte para llevar a cabo el desarrollo del proyecto. En este capítulo, se realiza un estudio de los avances producidos en la materia vistos hasta en la actualidad en el campo de la informática, así como todo lo relacionado con el estudio del PLN. Por otro lado, se analiza el contexto de la lectura fácil y de las diversas asociaciones y fuentes que contribuyen a la accesibilidad de la información. En este apartado se mencionan algunas herramientas existentes desarrolladas con este propósito.

En el Capítulo 3, se explica todo lo relacionado a la gestión del proyecto en cada una de las fases en que se puede dividir el mismo.

En el Capítulo 4, se muestra la funcionalidad, la configuración y las herramientas usadas para la generación de resúmenes en español, haciendo uso del servicio que ofrece Grafeno.

En el Capítulo 5, se desarrolla pormenorizadamente la funcionalidad, las herramientas usadas y la implementación del proceso, cuya función es la generación de un resumen simplificado y fácil de entender.

En el Capítulo 6, se realiza una evaluación del resultado en el desarrollo de la aplicación.

En el Capítulo 7, se muestran las conclusiones y líneas de trabajo futuro que se podrían desarrollar al término de este proyecto.

En el Capítulo 8, se muestra la bibliografía y las referencias tomadas en el desarrollo del mismo.

Finalizamos el documento del proyecto con un Apéndice, en el cual se detalla un manual de usuario para el uso de la aplicación Web **HAGRes** (Herramienta de Apoyo para la Generación de Resúmenes Fáciles de Leer en español).

## Capítulo 2:

# Estado del Arte

### 2.1 Introducción

Tal y como se menciona anteriormente, en este capítulo se muestra el estado actual de las tecnologías relacionadas con el proyecto, tanto las que nos servirán de apoyo, como las que están presentes en el resultado final.

Puesto que el objetivo de este proyecto está directamente relacionado con el Procesamiento de Lenguaje Natural (PLN), veremos su funcionalidad en la sección 2.2.

A continuación, en la sección 2.3 y dado que el proyecto tiene como objetivo mostrar un texto fácil de comprender, se muestra la definición y las reglas a seguir, así como las diferentes asociaciones existentes en este ámbito.

Por otro lado, analizamos las herramientas ya existentes de las que podremos hacer uso para el desarrollo del proyecto, para ello veremos la funcionalidad de cada una de estas herramientas en las secciones 2.4, 2.5 y 2.6.

Tras el estudio del estado del arte, podremos hacer uso de lo aprendido como punto de partida para la consecución del objetivo del proyecto.

## 2.2 Procesamiento de Lenguaje Natural (PLN)

El Procesamiento de Lenguaje Natural (más conocida en inglés como Natural Language Processing, PLN) nace en la década de los 60, como una subdisciplina de la Inteligencia Artificial (IA) y la Lingüística. Esta subdisciplina se encargará de investigar y formular mecanismos computacionalmente efectivos que faciliten la comunicación hombre-máquina permitiendo una comunicación mucho más fluida y menos rígida que los lenguajes formales.

Las primeras aplicaciones se centraron en la traducción automática (1940 – 1960), se basaron en la sustitución de palabra por palabra, dando resultados medianamente adecuados. Más adelante, se inició la investigación para resolver ambigüedades sintácticas y semánticas. La mayor parte del trabajo realizado en este período se centró en técnicas de análisis sintáctico.

A finales de los sesenta, se centra la investigación en la representación del significado, como resultado se diseñó el primer sistema de preguntas y respuestas basado en el lenguaje natural. Eliza [\[6\]](#) (Josep Weizenbaum, 1966), es un claro ejemplo de la comunicación entre la computadora y el usuario. Ésta representa el papel de un psicoanalista e imita sus respuestas típicas en una entrevista inicial.

En Europa surge el interés por crear programas para la traducción automática. Se crea el proyecto de investigación Eurotra [\[7\]](#), proyecto que tuvo como finalidad la traducción automática multilingüe. En Japón aparecen equipos dedicados a la creación de productos de traducción para su distribución comercial.

En los últimos años, el PLN [\[8\]](#) se viene desarrollando de muy diferentes formas y para receptores con distintas características. Tiene como objetivo desarrollar sistemas informáticos capaces de trabajar con el lenguaje verbal humano ya sea oral o escrito, con el fin de aportar utilidad al usuario. Las aportaciones que se han hecho han mejorado sustancialmente este procesamiento teniendo en cuenta aspectos relacionados con la sencillez a la hora de transmitir el mensaje, y el procesamiento de ingentes cantidades de información en formato texto con un grado de eficacia aceptable. Muestra de ello es la aplicación de estas técnicas como una componente esencial en los motores de búsqueda web, en las herramientas de traducción automática, o en la generación automática de resúmenes.

Uno de los problemas que intenta abordar el PLN es la **variación**, es decir, la posibilidad de expresar diferentes palabras o expresiones para comunicar una misma idea; y la ambigüedad lingüística, la cual se produce cuando una palabra o frase permite más de una interpretación. A continuación, se muestran algunos ejemplos que ilustran la repercusión de estos fenómenos en el proceso de recuperación de información:



- Ejemplo 1. A nivel morfológico.

*echa la bebida que sobre sobre la pila de la cocina, dijo llevando el sobre en la mano.*

La palabra **sobre** es ambigua morfológicamente ya que puede ser un sustantivo masculino singular, una preposición, y también la primera o tercera persona del presente de subjuntivo del verbo sobrar.

- Ejemplo 2. A nivel sintáctico.

*Juan vio a un niño con un telescopio en la ventana.*

La interpretación de la dependencia de los dos sintagmas preposicionales, con un telescopio y en la ventana, otorga diferentes significados a la frase: (1) Juan vio a un niño que estaba en la ventana y que tenía un telescopio, (2) Juan estaba en la ventana, desde donde vio a un niño que tenía un telescopio, y (3) Juan estaba en la ventana, desde donde miraba con un telescopio, y vio a un niño.

- Ejemplo 3. A nivel semántico.

*Maria dejó el periódico en el banco.*

El término banco puede tener dos significados en esta frase, (1) entidad bancaria y (2) asiento. La interpretación de esa frase va más allá del análisis de los componentes que forman la frase.

- Ejemplo 4. A nivel Pragmático.

*Ella le dijo que se fuera*

La interpretación de esta frase tiene diferentes incógnitas ocasionadas por la utilización de pronombres y adverbio: ¿quién habló?, ¿a quién?, ¿que se fuera de dónde?, ¿a dónde? Por tanto, para otorgar un significado a esta frase debe recurrirse nuevamente al contexto en que es formulada.

En la actualidad, la cantidad de información textual disponible ha crecido vertiginosamente en diferentes ámbitos como el académico, el laboral o el personal. Los correos electrónicos, documentos de trabajo, artículos científicos o publicaciones en redes sociales son grandes fuentes actuales de datos que se presentan en lenguaje natural. Esto genera un desafío, ya que el lenguaje es un tipo de dato no estructurado y que presenta ambigüedad. En este contexto, existe un creciente interés en mejorar la accesibilidad a la información y su explotación en diferentes entornos por parte de las empresas y organizaciones en general. Por todo esto, las aplicaciones de PLN se han vuelto de suma importancia en la actualidad.

Estos últimos procesos han dado lugar a diferentes herramientas, tal es el caso de Grafeno que actúa como servidor web de análisis y que usaremos como apoyo para el desarrollo del proyecto.

## 2.3 Accesibilidad a la información

Hoy en día, vivimos en una sociedad compleja, tecnificada y que aparentemente ofrece nuevas y mejores oportunidades de desarrollo personal y de bienestar social y calidad de vida. Una sociedad en la que *“una imagen vale más que mil palabras”*, con herramientas de orientación y comunicación desarrolladas en torno a la visión.

La accesibilidad se asociaba directamente con la necesidad de eliminar las barreras físicas como *“causante”* principal de los problemas. Sin embargo, este concepto ha ido evolucionando y en la actualidad se considera a la persona y a su entorno como un *“todo”* asumiendo que el medio en el que se desenvuelve debe diseñarse con el objetivo de asegurar sus derechos como ciudadano.

La diversidad del ser humano implica que cada individuo tenga unas capacidades diferentes o, dentro de las mismas, que unas estén más desarrolladas que otras. La capacidad de lectura es una de ellas y puede estar limitada por diversas circunstancias.

La accesibilidad universal, como derecho del ciudadano, es un concepto de reciente reconocimiento que está en proceso de integración dentro del conjunto social. Hablar de accesibilidad todavía sigue vinculándose a personas en sillas de ruedas por ser, posiblemente, una de las formas más visibles de la discapacidad en la que se pone de manifiesto la necesidad de tomar medidas para que ese derecho se convierta en una realidad.

Por lo tanto, al hablar de accesibilidad, estamos hablando de derechos y resulta necesario conocer el contexto normativo. En este sentido, podemos señalar, entre la normativa sobre accesibilidad para personas con discapacidad, las Normas de Naciones Unidas sobre la Igualdad de Oportunidades para las Personas con Discapacidad y, en nuestro país, la Ley de Igualdad de Oportunidades y Accesibilidad Universal. Estas normativas garantizan, en términos generales, la igualdad de oportunidades y la no exclusión.

En este contexto de accesibilidad universal, el acceso a los contenidos escritos significa, no sólo el acceso a la literatura, los diarios o las enciclopedias y libros de texto, sino también a la legislación, a los documentos administrativos, informes médicos, contratos y cualquier otro texto de la vida cotidiana [\[9\]](#).

El acceso a los textos se puede lograr mediante una herramienta que permita una lectura fácil de los mismos de forma que se facilite la comprensión lectora para personas con dificultad e incluso a aquellas que no tengan el hábito de leer. Esta herramienta pretende ser una solución para facilitar el acceso a la información, la cultura y la literatura, debido a que es un derecho fundamental de las personas, que son iguales en derechos, con independencia de sus capacidades [\[10\]](#).

Por todo lo anteriormente mencionado, la accesibilidad se ha convertido en uno de los mayores retos y es un indicador claro del progreso y del desarrollo social alcanzado.

### 2.3.1 La Lectura Fácil

La Lectura Fácil se define como una técnica de redacción y publicación que permite la comprensión textual a personas con dificultades lectoras. Desde hace más de 40 años empezó a aplicarse en Suecia y Estados Unidos para múltiples tipos de textos, tanto literarios, como jurídicos o informativos. Este tipo de textos en lectura fácil, no solo irá dirigido a gente con discapacidad, sino también a personas con baja formación cultural o problemas de tipo social.

La definición de lectura fácil se asentó en 1997, con la publicación de “*Directrices para materiales de lectura fácil*” de la Federación Internacional de Asociaciones de bibliotecarios y librerías (IFLA), que se revisó posteriormente en 2010.

En España se creó la asociación de Lectura Fácil, integrada por profesionales de la docencia y otras actividades. Se encarga de impulsar la producción, edición y difusión de los materiales de Lectura Fácil (LF), siguiendo las directrices internacionales de la IFLA [\[11\]](#).

No obstante, en los últimos años han surgido varios textos que toman como referencia a la IFLA e Inclusión Europea, pero añaden matices y modificaciones. Este hecho ha justificado la necesidad de una reorganización y sistematización para aclarar algunos aspectos. Cabe señalar que algunas de esas pautas han ido acompañadas de estudios más detallados con personas con discapacidad intelectual, que han permitido crear unas directrices para las pruebas de contraste de textos.

Con respecto a la situación actual de la metodología de LF, algunas características que se señalan dentro del método son: [\[12\]](#):

- Utiliza un lenguaje simple y directo.
- Estructura el texto de manera clara y coherente.
- Evita los tecnicismos, las abreviaturas y las iniciales.
- Expresa una sola idea por frase.
- Evita en lo posible conceptos abstractos.
- Utiliza fotografías, gráficos o símbolos de apoyo al texto.

A la hora de elaborar un documento de fácil lectura, es conveniente tener en cuenta lo siguiente:

- Es necesario consultar a algunas personas con discapacidad intelectual durante todo el proceso de elaboración de los textos.

- Se debe mantener cierta flexibilidad a la hora de poner en práctica las propuestas o directrices.
- Hay que tener en cuenta, antes de comenzar, el grupo al que va dirigida la información y la finalidad principal de la publicación.
- Pueden darse dos situaciones distintas:
  - Que se dispone de un texto base que hay que hacer accesible.
  - Que se genera un texto completamente nuevo.

En cuanto a las normas generales de **elaboración**, la Asociación Europea ILSMH plantea, entre otras, las siguientes:

- Utilizar frases cortas.
- Expresar una idea en cada frase.
- No utilizar metáforas o comparaciones que puedan ser confusas.
- Evitar frases en negativa
- Evitar los conceptos abstractos.
- Emplear vocablos cortos relativos al lenguaje cotidiano hablado.
- Hacer uso de ejemplos prácticos.
- Dirigirse a los lectores de manera respetuosa.
- Contar las historias con orden, según ocurren las cosas.
- Utilizar el orden sintáctico más básico y simple.
- Emplear preferentemente la voz activa frente a la pasiva.
- No dar por asumidos conocimientos previos sobre el tema en cuestión.
- Ser sistemático al utilizar las palabras.
- Elegir signos de puntuación sencillos.
- No emplear el subjuntivo.
- Tener cuidado con el lenguaje.
- Evitar el uso de jergas, abreviaturas e iniciales.
- Tener cuidado con el uso de números.
- No emplear palabras de otro idioma.

- Evitar el uso de referencias.

Las recomendaciones más importantes a tener en cuenta en lo que se refiere al **diseño** son:

- Utilizar como máximo dos tipos de letras dentro del mismo documento.
- El tamaño de la letra ha de ser grande o configurable si el formato es electrónico.
- No utilizar mayúsculas ni cursivas. Usar la negrilla o el subrayado para enfatizar palabras o frases.
- No justificar el texto a la derecha.
- No poner dibujos como fondo de un texto.
- Intentar utilizar una sola línea para cada oración.
- Evitar separar los elementos constitutivos de la oración, de modo que ésta quede siempre dentro de una misma página.
- No incluir demasiada información en una página.
- No usar nunca la impresión invertida.
- Utilizar colores para los dibujos.
- No usar guiones para separar palabras largas en el margen derecho del texto.

Para finalizar, y para constatar la importancia y necesidad de la Lectura Fácil, se establecen los motivos por los cuales se hace deseable el desarrollo de esta técnica:

- Facilita el acceso a la lectura y a la información como derecho y necesidad social.
- Permite compartir ideas, pensamientos y experiencias.
- Facilita el acceso a textos con exceso de tecnicismos, sintaxis compleja y presentación poco clara.
- Permite a más del 30% de la población con dificultades lectoras, al acceso a la información.

### 2.3.2 Accesibilidad de contenidos Web

La Accesibilidad Web nace de la iniciativa de WAI (Web Accessibility Initiative), se trata del **W3C**, un grupo internacional e independiente que define los protocolos y estándares para la web. Una de las principales iniciativas del W3C es el desarrollo de normas de accesibilidad. El objetivo de la Iniciativa para la Accesibilidad Web (Web Accessibility Initiative, WAI) es desarrollar los estándares de accesibilidad. Los grupos de trabajo WAI se

encargan de desarrollar las normas de accesibilidad para los navegadores Web, para las herramientas de autor, de navegación y contenido web.

Para lograr la Accesibilidad, se han desarrollado diferentes pautas o guías en las que se explican cómo lograr que el diseño sea accesible en las páginas Web, reduciendo de esta forma las barreras del acceso a la información. **WCAG** consiste en 14 pautas [\[13\]](#) o principios que proporcionan soluciones de diseño y ejemplos comunes en las que el diseño de una página puede producir problemas de acceso a la información. Además de las pautas o principios mencionados tenemos los llamados “*Criterios de éxito*”, cada uno de los cuales se indican en una sola palabra: Perceptible, Operable, Comprensible y Robusto.

Hablar de Accesibilidad Web es hablar de un acceso universal a la Web, independientemente del tipo de hardware, software, infraestructura de red, idioma, cultura, localización geográfica y capacidades de los usuarios. A continuación, mencionamos algunas de las definiciones encontradas sobre Accesibilidad Web.

Para el grupo de trabajo **WAI** (Iniciativa para la Accesibilidad de la Red), la Accesibilidad Web significa “*que personas con algún tipo de discapacidad van a poder hacer uso de la Web. En concreto, al hablar de accesibilidad Web se está haciendo referencia a un diseño Web que va a permitir que estas personas puedan percibir, entender, navegar e interactuar con la Web.*”

Según **Inteco** (Instituto Nacional de Tecnologías de la Comunicación) se puede definir la accesibilidad como “*la posibilidad de que un sitio o servicio Web pueda ser visitado y utilizado de forma satisfactoria por el mayor número posible de personas, independientemente de las limitaciones personales que tengan o de aquellas limitaciones que sean derivadas de su entorno.*”

Para **AENOR** (Asociación Española de Normalización y Certificación) hablar de Accesibilidad Web es hablar de “*un acceso universal a la Web, independientemente del tipo de hardware, software, infraestructura de red, idioma, localización geográfica y capacidades de los usuarios*” (AENOR).”

La web ofrece muchas oportunidades a las personas con discapacidad que no están disponibles a través de cualquier otro medio, ofrece independencia y libertad. Sin embargo, si un sitio web no se crea con la accesibilidad web necesaria, puede excluir a un segmento determinado de la población. A medida que las organizaciones y los diseñadores tomen conciencia y pongan en práctica la accesibilidad, garantizarán que su contenido puede ser accedido por una población más amplia.

Por lo tanto, un sitio Web es accesible cuando puede ser utilizado de forma eficaz para el mayor número de personas, independientemente de sus conocimientos, edad, o capacidades personales e independientemente de los dispositivos involucrados para acceder a la web.

### 2.3.3 Simplificación de textos

Se entiende como *simplificación de textos* al proceso de transformar un texto en otro equivalente que resulte más fácil de leer y probablemente de entender por un determinado grupo de personas. En este proceso, las oraciones complejas se dividen en oraciones más simples y el vocabulario complejo se reemplaza por un vocabulario más sencillo, manteniendo la información y el significado original.

La Simplificación de texto intenta abordar tres problemas:

- Uno de los problemas es identificar la dificultad de un texto a fin de decidir si es apropiado para una audiencia determinada. En general esto significa clasificar el texto según su nivel de legibilidad.
- El segundo problema consiste en simplificar la sintaxis de las oraciones. Construcciones como las subordinaciones o las coordinaciones son generalmente simplificadas en oraciones independientes.
- El tercer problema consiste en simplificar el vocabulario del texto. En general las palabras poco frecuentes o que se consideran complicadas son reemplazadas por palabras sinónimas más comunes.

La simplificación del texto es una tecnología del PLN que conforme vaya madurando podría ser utilizada para producir textos adaptados a las necesidades específicas de usuarios con limitaciones.

La mayoría de las investigaciones realizadas en el área de la simplificación automática de texto ha tratado el idioma inglés, no obstante, en los últimos años se ha ido desarrollando en España el proyecto **Simplext** (Sistema automático de transformación de contenidos en textos de fácil lectura) y **Able-to-Include** [\[14\]](#). Estos proyectos tienen como objetivo desarrollar un sistema automático de simplificación de textos no adaptados.

### 2.3.4 Proyecto Simplext y Able-to-Include

El antecedente del proyecto se encuentra en el *proyecto «Lectura fácil: elaboración y difusión de materiales y análisis de la experiencia de usuario»*, promovido por la Fundación **ONCE** y financiado por el Plan Avanza del Ministerio de Industria, Comercio y Turismo en 2008. Esta idea tenía como objetivo desarrollar una web de noticias en lectura fácil.

El resultado fue el portal [www.noticiasfacil.es](http://www.noticiasfacil.es) (Figura 2) [15], asociado al portal de información social **Discapnet** (Figura 3) [16]. Si bien fue un primer paso para avanzar en la adaptación de la información para personas con discapacidad intelectual, se ha visto que la selección de noticias tiende a temas de escasa relevancia o anecdóticos y la extensión es excesivamente breve para aportar una verdadera información.

El proyecto Simplext y Able-to-Include da un paso más, ya que propone el desarrollo de un sistema general para la simplificación automática de textos, basándose en principios de Lectura Fácil y en la aplicación de técnicas robustas de PLN, así como en las tendencias tecnológicas en accesibilidad para la obtención de un producto de apoyo omnipresente e interoperable.

La metodología adoptada en el proyecto Simplext se basa en la creación de un corpus de textos periodísticos y sus simplificaciones manuales con el fin de realizar un estudio que permitiera discernir qué manipulaciones serían necesarias para obtener una simplificación automática correcta. Las simplificaciones manuales fueron producidas por el grupo de investigación **DILES** de la Universidad Autónoma de Madrid. Los diferentes estudios del corpus realizado (Stefan Bott & Saggion, 2011) (Drndarevic & Saggion, 2012) proponen dos tareas principales:

- **Simplificación sintáctica**, encargada de transformar oraciones largas y complejas en oraciones simples e independientes, segmentando construcciones subordinadas y coordinadas.
- **Simplificación léxica**, encargada de reemplazar palabras difíciles por sinónimos simples usando una base de datos psicolingüística y un diccionario de sinónimos (WordNet).

También se estudiaron operaciones de re-escritura que hacen que los textos simplificados resulten más fáciles de entender. Algunos ejemplos de simplificaciones en el corpus Simplext pueden apreciarse a continuación.

- **Original:** Abre en Madrid su primera sucursal el mayor banco de China y del Mundo.
- **Simplificación:** El banco más importante de China y del mundo abre una oficina en Madrid.



Esta simplificación utiliza la palabra **oficina** en lugar de la palabra **sucursal** que es mucho más específica y por lo tanto más compleja. Además, la frase simplificada utiliza el orden canónico de los elementos en la frase en español (**sujeto-verbo-objeto**) a diferencia del orden utilizado en la oración original.

Para desarrollar el componente de simplificación sintáctica en Simplext se adoptó una técnica basada en reglas que operan sobre la estructura sintáctica de dependencias de las frases (Stefan Bott, Saggion, & Mille, 2012).

- La primera regla identifica y construye de relativo, participio, gerundio, y varios casos de coordinación sintáctica e identifica en las oraciones “*puntos de corte*” con el fin de separar las oraciones en sus componentes.
- La segunda regla copia los elementos de la frase original a los componentes más simples.
- La tercera regla decide el orden en el que las diferentes frases serán presentadas. Finalmente se utiliza un sistema de generación de frases para producir las oraciones correctas.

Con respecto al componente de simplificación léxica, se implementó el sistema LexSiS que utiliza técnicas robustas de PLN. LexSiS realiza un área de desambiguación de sentidos de las palabras en contexto para escoger sinónimos apropiados de las mismas.

Una vez desambiguada una palabra, se selecciona el sinónimo más simple para sustituir la palabra original utilizando un criterio de simplicidad. Para realizar estas tareas, LexSiS cuenta con un diccionario de sentidos y sinónimos que está disponible libremente y utiliza información sobre frecuencia y longitud para seleccionar el sinónimo más simple (Stefan Bott, Saggion, et al., 2012) (Saggion, Bott, & Rello, 2013).

El tercer componente de Simplext es un sistema de re-escritura que se encarga de transformaciones muy precisas que los simplificadores humanos aplican al texto y que no pueden ser tratadas por los componentes anteriores.

Los tres componentes, conjuntamente con las herramientas básicas de análisis del texto en español (etiquetadores morfosintácticos, parsers, reconocedores de entidades nombradas), han sido integrados en el motor de simplificación Simplext.

Simplext, tiene como objetivo crear herramientas que permitan la conversión a lectura fácil de textos. Esta herramienta debe permitir la lectura de textos en base a la reducción de la complejidad léxica y sintáctica de los textos en el que se detecta las palabras y oraciones más complejas y generar vocabulario más simple y oraciones más cortas.

Uno de los desarrollos tecnológicos del proyecto es un portal Web de simplificación de textos, desarrollada por el grupo de Tratamiento Automático del Lenguaje Natural (**TALN**) [\[17\]](#) de la Universitat Pompeu Fabra (UPF).



*Figura 1: Portal web Simplext*

La arquitectura de Simplext es una arquitectura orientada a servicios que desarrolla aún más la idea de ofrecer el servicio de simplificación. La razón de este enfoque es que una persona con un dispositivo adaptado pueda, por ejemplo, leer contenidos digitales en teléfonos móviles, los cuales deben ser capaces de recibir contenidos digitales de fácil lectura, como RSS o prensa digital.

El servicio web de simplificación es el componente de software que permitirá publicar el motor de simplificación utilizando el software SaaS (Software as a Service) como modelo de entrega de software. Para desarrollar esta capa de servicios, lo que permitirá entregar las funcionalidades antes mencionadas, se planea una arquitectura basada en SOA y REST (Representational State Transfer). Se accede a las capacidades del motor de simplificación utilizando identificadores estándar (URI), donde cada solicitud puede hacerse de forma independiente, atómica, sin estado.

Otras de las herramientas que podemos encontrar actualmente son:

**Dilofácil.** (Figura 4), es un proyecto especializado en la creación de textos accesibles para cualquier persona, DiloFácil promueve la investigación para la mejora de la metodología y participa en eventos especializados para difundir esta herramienta de facilitación lectora.

**PorSimples** (Figura 5), El objetivo principal de este proyecto fue desarrollar tecnologías de procesamiento de lenguaje natural (PLN) relacionadas con la adaptación de texto para promover la inclusión digital y la accesibilidad para las personas con bajos niveles de alfabetización. Actualmente solo se encuentra disponible en Portugués.

- **Noticia Fácil**



Figura 2: Portal web Noticia Fácil

- **Discapnet**



Figura 3: Portal web Discapnet

- **Dilofácil**



Figura 4: Portal web diloFácil

- **PorSimples**

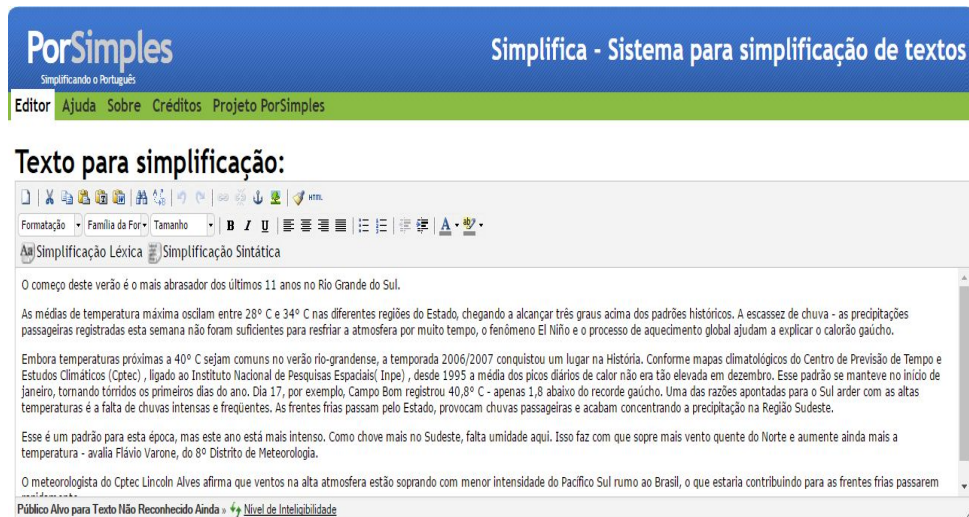


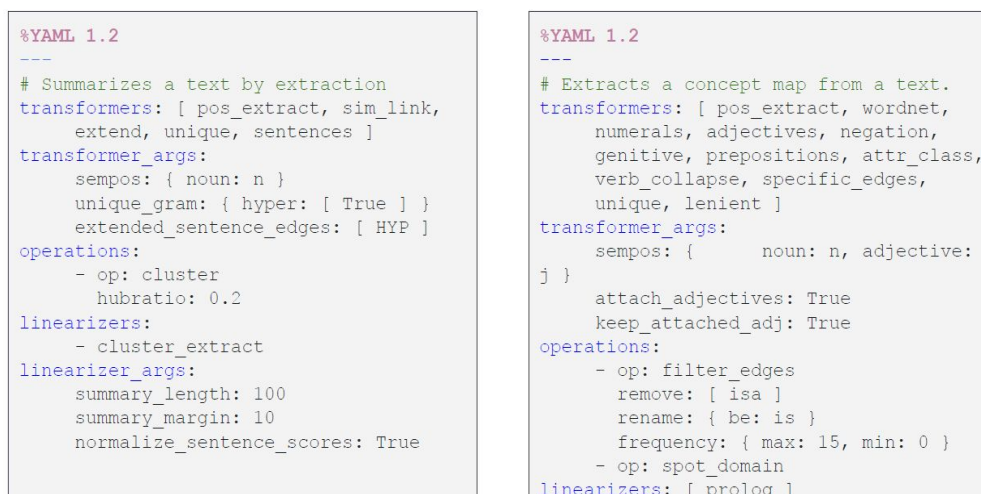
Figura 5: Portal web PorSimples

## 2.4 Herramientas de apoyo

### 2.4.1 Grafeno

Grafeno fue desarrollado por Antonio F. G. Sevilla y por el grupo de investigación **NIL** (*Natural Interaction based on Language* o Interacción Natural basada en el Lenguaje), un grupo de investigadores y profesionales de la UCM, cuyos intereses van enfocados al desarrollo de interfaces basadas en el Lenguaje Natural, con el objetivo de investigar y desarrollar nuevas tecnologías que puedan ser utilizadas en aplicaciones prácticas.

Grafeno es una librería multifuncional desarrollada en Python y funciona por línea de comandos de Linux. Cuenta con una opción de ayuda que nos indica que necesita para su funcionamiento un texto de entrada y las operaciones que se van a realizar sobre el texto. Estas operaciones pueden realizarse de forma individual proporcionando un transformador y un linearizer, o bien pueden realizarse de forma conjunta indicando un fichero de configuración o yaml. Cuenta con fichero yaml predefinidos para ciertas operaciones, como la de generara resúmenes por extracción, y se pueden usar sin o con poca modificación tal y como se muestra en la Figura 6.



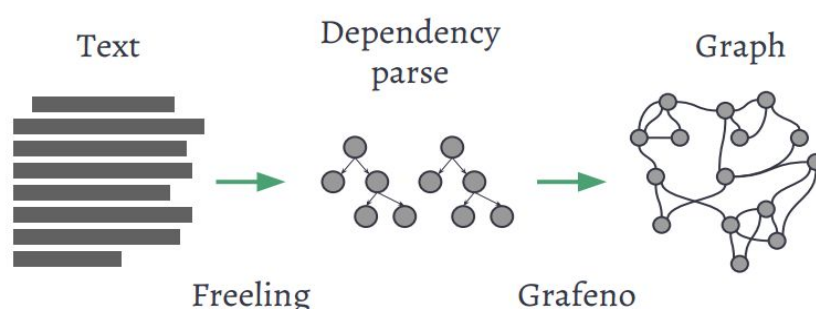
```
%YAML 1.2
---
# Summarizes a text by extraction
transformers: [ pos_extract, sim_link,
               extend, unique, sentences ]
transformer_args:
  sempos: { noun: n }
  unique_gram: { hyper: [ True ] }
  extended_sentence_edges: [ HYP ]
operations:
  - op: cluster
    hubratio: 0.2
linearizers:
  - cluster_extract
linearizer_args:
  summary_length: 100
  summary_margin: 10
  normalize_sentence_scores: True
```

```
%YAML 1.2
---
# Extracts a concept map from a text.
transformers: [ pos_extract, wordnet,
               numerals, adjectives, negation,
               genitive, prepositions, attr_class,
               verb_collapse, specific_edges,
               unique, lenient ]
transformer_args:
  sempos: { noun: n, adjective:
j }
  attach_adjectives: True
  keep_attached_adj: True
operations:
  - op: filter_edges
    remove: [ isa ]
    rename: { be: is }
    frequency: { max: 15, min: 0 }
  - op: spot_domain
linearizers: [ prolog ]
```

Figura 6: Scripts de configuración Yaml.

El funcionamiento de Grafeno está basado en el análisis de dependencias. Dado un texto de entrada, mediante clustering se forman grafos de dependencias de distintos tamaños que representan los temas principales y los conceptos más referenciados.

Esta librería fue desarrollada con la idea fundamental de resumir el texto basándose en gráficos conceptuales. Para llevar a cabo esta operación, en primer lugar, se usa un enfoque extractivo, y en segundo lugar se centra en una descripción más semántica del texto.



*Figura 7: Grafo de extracción automática*

Para enriquecer los grafos de información conceptual, Grafeno hace uso de herramientas existentes y bases de conocimiento tales como WordNet y FreeLing, que se detallan en la sección 2.4.2 y 2.4.3 respectivamente.

Existen multitud de proyectos fin de carrera que hacen uso de esta librería, tal es el caso de “Generación de Lenguaje Natural a partir de Grafos Semánticos”, al que haremos referencia en el capítulo 3.

Está librería sólo entiende inglés, pero está diseñado para ser independiente del lenguaje. Por supuesto la independencia del lenguaje dependerá de la semántica en cuestión, por otro lado, Grafeno dependerá por completo de la especificidad de los módulos y reglas utilizadas.

El texto sin formato y depurado, se ejecuta a través de un analyzer pipeline, quien analiza morfológicamente las palabras y extrae un árbol de dependencias de cada frase. Para el análisis de la dependencia, Grafeno hace uso de FreeLing que es una librería rápida y fiable, con configuración pipeline para varios idiomas. No obstante, se pueden usar otras herramientas de análisis sintáctico para esta etapa del procesamiento (por ejemplo, MaltParser).

Una vez finalizada el proceso del analizador lingüístico, el propio procesamiento de grafeno se inicia. En primer lugar, las palabras morfológicamente analizadas se transforman en los nodos semánticos, con el "concept" de referencia, donde las relaciones de dependencia son procesadas, algunos de ellos se transforman en aristas semánticas con un correspondiente "functor".

Una vez creado los nodos y los edges, se podrá realizar el procesamiento adicional, donde se pueden combinar, eliminar e incluso crear nuevos edges. El resultado final es una función de textos de entrada.



En la siguiente Figura 8 se muestra los módulos que se han utilizado y el orden en el que se ha llevado a cabo.

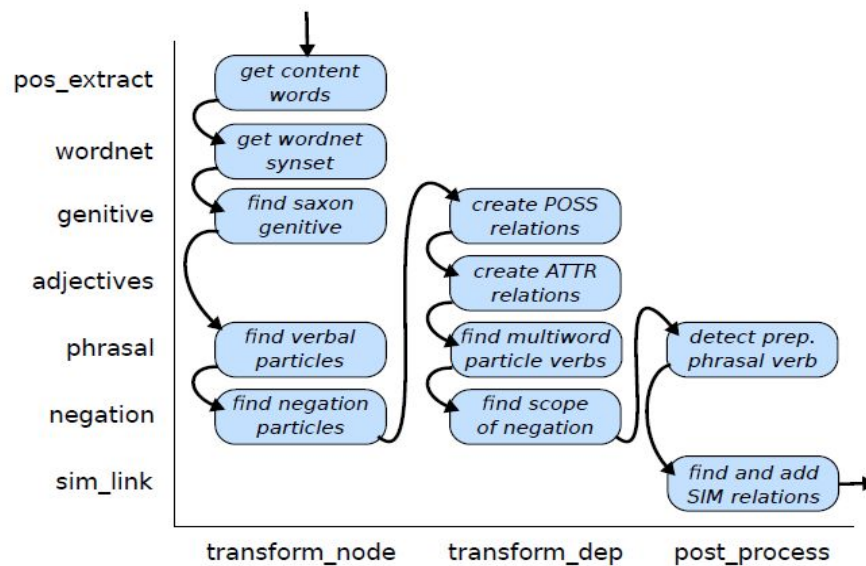


Figura 8: Flujo de ejecución entre módulos y pasos de procesamiento en un transformador compuesto.

Durante la primera etapa se realiza la extracción del grafo, se configura en los valores **transformers** y **transformer\_args**, especificando los módulos y los parámetros que se utilizarán. Se extraen los módulos **pos\_extract** y los parámetros **sempos**, los módulos denominados **unique** aseguran que para cada concepto solo hay un nodo, incluso en el caso de que apareciera varias veces en el texto; **extend** añade información WordNet en forma de estructura hypernym.

Otro módulo denominado **sim\_link** relaciona los conceptos de acuerdo a la similitud léxica. **sentences**, donde cada frase es representada por un nodo sirven para generar el resumen final. Una vez creado el grafo se realiza una operación de agrupamiento.

Como última fase, el módulo utilizado es el cluster extract un generador de resumen de forma extractiva que selecciona las frases más próximas y relacionadas con el tema principal (cluster) del texto.

## 2.4.2 FreeLing

FreeLing es una librería de código abierto, bajo una licencia GNU (General Public License), que proporciona servicio de análisis multilingüe, para varios idiomas. Este proyecto se inició en el centro TALP de la UPC y actualmente es liderado por Lluís Padro [\[18\]](#).

FreeLing fue creado con el objetivo de poner a disposición de la comunidad los resultados de la investigación realizada en el grupo de investigación de PLN.

FreeLing es una librería desarrollada en C++, que proporciona funcionalidades de análisis lingüístico (análisis morfológico, detección de entidades nombradas, etiquetado PoS, análisis sintáctico, desambiguación de palabras, etiquetado de roles semánticos, etc.) para una variedad de idiomas (inglés, español, portugués, italiano, Alemán, ruso, catalán, gallego, croata, esloveno, entre otros), tal y como se muestra en la Figura 9.

Este proyecto, se estructura como una librería que puede ser llamada desde cualquier aplicación de usuario que requiera servicios de análisis de lenguaje y sobre la cual se puedan desarrollar potentes aplicaciones de PLN, y orientado a facilitar la integración con las aplicaciones de niveles superiores de los servicios lingüísticos que ofrece.

FreeLing está diseñado para ser modular y mantener los datos lingüísticos separados del código. Por lo tanto, la mayoría de los módulos se pueden adaptar a un nuevo idioma simplemente reemplazando un archivo de configuración o proporcionando un archivo con reglas específicas para ese idioma.

FreeLing procesa el texto y crea estructuras de datos que representan los objetos lingüísticos de ese texto, los cuales son elementos como palabra, PoS-tag, relaciones, parse\_tree, etc. Así, los módulos de procesamiento FreeLing son capaces de convertir un texto en una colección de objetos lingüísticos.



La siguiente tabla muestra los servicios de análisis disponibles para cada lengua.

	as	ca	cy	de	en	es	fr	gl	hr	it	nb	pt
Tokenization	X	X	X	X	X	X	X	X	X	X	X	X
Sentence splitting	X	X	X	X	X	X	X	X	X	X	X	X
Number detection		X		X	X	X	X	X		X		X
Date detection		X		X	X	X	X	X				X
Morphological dictionary	X	X	X	X	X	X	X	X		X	X	X
Affix rules	X	X	X	X	X	X	X	X		X	X	X
Multiword detection	X	X	X		X	X	X	X		X		X
Basic named entity detection	X	X	X		X	X	X	X		X		X
B-I-O named entity detection		X			X	X		X				X
Named Entity Classification		X			X	X						X
Quantity detection		X			X	X		X				X
PoS tagging	X	X	X	X	X	X	X	X		X	X	X
Phonetic encoding					X	X						
WN sense annotation		X			X	X	X	X	X			
UKB sense disambiguation		X			X	X	X	X	X			
Shallow parsing	X	X			X	X		X				X
Full/dependency parsing	X	X			X	X		X	X			
Semantic Role Labelling		X		X	X	X						
Coreference resolution					X	X						

Figura 9: Servicios de análisis disponible para cada lengua

Este diagrama UML de la Figura 10, muestra las clases en las que toda la información lingüística es almacenada por los módulos de procesamiento. La aplicación que llama a FreeLing puede acceder posteriormente a estas estructuras de datos para recuperar los resultados de los analizadores.

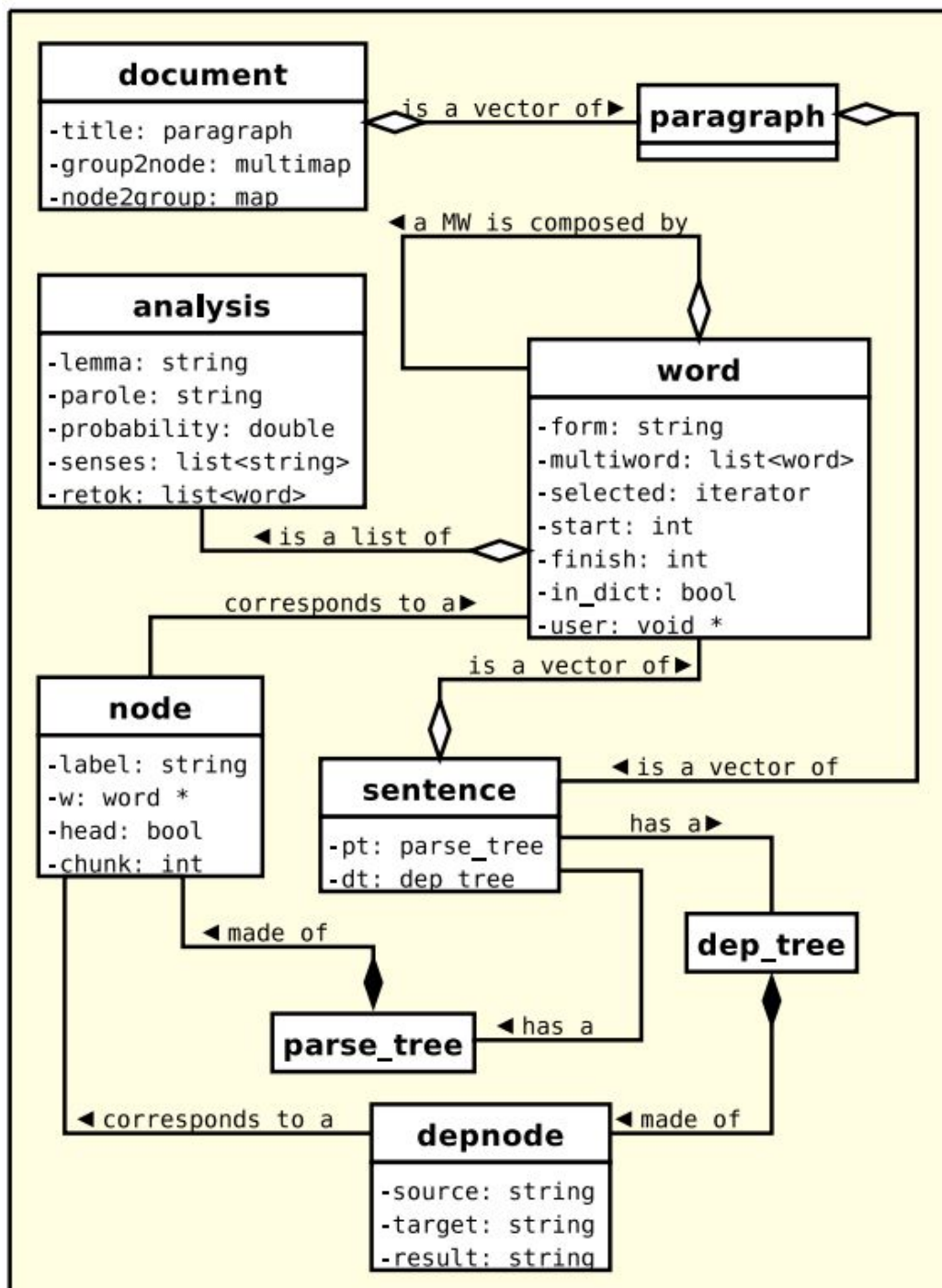


Figura 10: Módulos de procesamiento FreeLing

### 2.4.3 WordNet

WordNet es una base de datos léxica, de código abierto, donde los sustantivos, verbos, adjetivos y adverbios se agrupan en conjuntos de sinónimos cognitivos (synsets) cada uno de los cuales expresa un concepto distinto y proporciona definiciones cortas y generales. Los synsets también están conectadas entre sí a través de relaciones léxicas como hypernym o hyponym, cuyo objetivo es la creación de una estructura rica y útil para la lingüística computacional y el PLN [19].

WordNet se puede consultar directamente en su web y utilizarlo. En la Figura 11 podemos observar la interfaz web que proporcionan. También podemos integrarlo en un sistema, para ello lo más sencillo es instalarlo y utilizarlo de forma local.

The screenshot displays the WordNet Search interface. At the top, there's a header with the title 'WordNet Search - 3.1' and links to the 'WordNet home page', 'Glossary', and 'Help'. Below this, a search bar contains the word 'read' and a 'Search WordNet' button. Under the search bar, there are 'Display Options' with a dropdown menu set to '(Select option to change)' and a 'Change' button. A key explains the options: 'S:' for Show Synset (semantic) relations and 'W:' for Show Word (lexical) relations. Below the key, it shows 'Display options for sense: (gloss) "an example sentence"'. The results are categorized under 'Noun' and 'Verb'. Under 'Noun', there is one entry: 'S: (n) read (something that is read) "the article was a very good read"'. Under 'Verb', there are ten entries, each with a part of speech (S or V), a list of related words, and an example sentence. The entries are: 1. S: (v) read (interpret something that is written or printed) "read the advertisement"; "Have you read Salman Rushdie?"; 2. S: (v) read, say (have or contain a certain wording or form) "The passage reads as follows"; "What does the law say?"; 3. S: (v) read (look at, interpret, and say out loud something that is written or printed) "The King will read the proclamation at noon"; 4. S: (v) read, scan (obtain data from magnetic tapes or other digital sources) "This dictionary can be read by the computer"; 5. S: (v) read (interpret the significance of, as of palms, tea leaves, intestines, the sky; also of human behavior) "She read the sky and predicted rain"; "I can't read his strange behavior"; "The fortune teller read his fate in the crystal ball"; 6. S: (v) take, read (interpret something in a certain way; convey a particular meaning or impression) "I read this address as a satire"; "How should I take this message?"; 7. S: (v) learn, study, read, take (be a student of a certain subject) "She is reading for the bar exam"; 8. S: (v) read, register, show, record (indicate a certain reading; of gauges and instruments) "The thermometer showed thirteen degrees below zero"; "The gauge read 'empty'"; 9. S: (v) read (audition for a stage role by reading parts of a role) "He is auditioning for 'Julius Caesar' at Stratford this year"; 10. S: (v) read (to hear and understand) "I read you loud and clear!"; 11. S: (v) understand, read, interpret, translate (make sense of a language) "She understands French"; "Can you read Greek?"

Figura 11: Interfaz web WordNet 3.1

## Capítulo 3:

# Generación del Proyecto

### 3.1 Fase Inicial

#### 3.1.1 Objetivos

En esta fase inicial del proyecto nos enfocamos en generar una herramienta que sirviera de apoyo a la hora de redactar el resumen de un texto medianamente extenso.

Como punto de partida contamos por un lado con un proyecto de Máster orientado a realizar una tarea parecida. La idea de este proyecto de máster es generar lenguaje natural a partir de grafos semánticos, usando para ello herramientas lingüísticas tales como WordNet [\[19\]](#), ConceptNet 5 [\[20\]](#), Textifier [\[21\]](#), etc. Para generar el lenguaje natural necesita recibir como datos de entrada un grafo formado por tripletas en formato JSON que representan relaciones entre conceptos.

Por otro lado, contamos con Grafeno, una herramienta multifuncional desarrollada en Python dentro de un entorno virtual de Ubuntu. Grafeno recibe como datos de entrada el texto sobre el que queremos trabajar, un linearizer y un transformador con los que indicamos qué operación queremos realizar sobre el texto de entrada. También puede recibir en lugar del linearizer y del transformador, un fichero de configuración que contiene un listado de linearizers y transformadores que indican el conjunto de instrucciones que queremos realizar sobre el texto, para así poder obtener resultados más complejos.

Haciendo uso de esta funcionalidad de Grafeno, que se sirve además de FreeLing para analizar las palabras, nuestro objetivo era combinar la herramienta Grafeno con el proyecto de máster de tal modo que las tripletas que devolviera Grafeno como salida, fueran los datos de entrada necesarios para el proyecto de máster en la generación del texto en lenguaje natural.

### 3.1.2 Desarrollo

De forma resumida, dado que tanto el funcionamiento interno de Grafeno como el del proyecto de máster nos eran completamente ajenos, comenzamos estudiando cómo estaban desarrollados y qué necesitaban para funcionar.

Por un lado, el proyecto de máster recibe como datos de entrada un conjunto de tripletas que representan el grafo semántico y que se procesan hasta obtener un conjunto de tripletas bien estructurado. Pasos:

- Se cambian algunas relaciones por otras más genéricas, para conseguir que haya más conceptos de cada relación.
- Se juntan las tripletas con el mismo concepto sujeto y la misma relación.
- Se ordenan las tripletas de mayor a menor cantidad de conceptos objeto, priorizando las que tienen el concepto sujeto como entidad principal.

Una vez procesadas todas las tripletas comienza la fase de generación o bien de un texto simple o bien de un texto más elaborado:

- Generación de texto simple: a partir del conjunto de tripletas previamente procesado se genera el texto extrayendo, para cada triplete, el concepto sujeto, la relación y el concepto objeto, de este modo el texto tiene la estructura de sujeto verbo predicado.
- Generación de texto elaborado: tras localizar las tripletas directamente relacionadas con el tema principal, se buscan en ellas las relaciones de tipo Es\_Un para realizar una descripción del concepto del que se habla.

Ya que el proyecto de máster necesitaba como datos de entrada un conjunto de tripletas para generar un texto, quisimos aprovechar la funcionalidad de Grafeno de, a partir de un documento de texto, generar las tripletas que representan los datos más importantes de ese texto. Así, las tripletas que ofreciera Grafeno como datos de salida servirían como datos de entrada para el proyecto de máster y éste finalmente generaría el resumen a partir de éstas.

Como primera aproximación y por probar que era posible fusionar ambos proyectos, pasamos manualmente las tripletas devueltas por Grafeno al formato de las tripletas del proyecto de máster y las guardamos en un fichero de texto que sería la entrada del proyecto de máster. Observamos que el módulo de procesamiento de tripletas del proyecto de máster estaba limitado a un conjunto finito de relaciones, es decir, que solo reconocía tripletas cuya relación (el verbo) eran de tipo “to be”, “eat” o “have”. Siendo así no pudimos contar con la funcionalidad de este proyecto de máster puesto que no nos serviría para generar resúmenes de texto que trataran sobre cualquier tema.

### **3.1.3 Resultados y discusión crítica**

Al final no pudimos completar la fusión de proyectos. Por un lado, llevaría bastante tiempo hacer que Grafeno devolviera las tripletas en un formato comprensible para el proyecto de máster y por otro lado, pensamos que resultaría más útil y práctico poder generar los resúmenes en español, pues se trata de un idioma con el que estamos más cómodos.

El motivo principal por el cual descartamos esta opción se debe a que el proyecto de máster está limitado en cuanto al texto que puede generar. Para el procesamiento de las tripletas se tiene en cuenta el verbo de la relación, y los únicos verbos que podía soportar son “to be”, “eat” y “have”. Es decir, para poder darle utilidad tendríamos no solo que adaptar el proyecto para que trabajara también con texto en español, sino que además habría que considerar todos los verbos y posibles conectores que tiene nuestro idioma. Dado que esta opción no nos pareció viable, optamos por avanzar a otra etapa.

## **3.2 Fase Final**

En esta segunda fase conseguimos desarrollar la herramienta que servirá de apoyo para la redacción de resúmenes y se desarrolla como una aplicación gráfica para facilitar su uso. Este desarrollo de la herramienta se divide en dos partes: la generación de resúmenes y la simplificación de textos.

Para la parte de generación de resúmenes empezamos modificando la configuración Grafeno para que pudiera tratar con textos en español. Luego buscamos un modo de facilitar el acceso al resumen y a su redacción, dando al usuario la posibilidad de decidir cómo de extenso quiere que sea el resumen.

En la parte de la simplificación, dado un texto se analiza y se simplifica eliminando aquella información excesiva o no necesaria para facilitar su comprensión. Además se sustituyen por sinónimos más sencillos aquellas palabras específicas de un ámbito o campo concreto. Esta parte a pesar de estar directamente ligada con la primera, tiene una funcionalidad independiente, es decir que puede usarse directamente sobre un texto cualquiera, no necesariamente el texto resumido.

## Capítulo 4

# Generación de Resúmenes

### 4.1 Objetivos

En esta etapa nuestro objetivo era aprovechar la funcionalidad de Grafeno para producir resúmenes por extracción y así generar resúmenes de documentos de texto escritos en español. Esto quiere decir que buscamos el modo de adaptar una nueva versión de Grafeno para que pudiera trabajar con texto en español. Una vez completada la adaptación, se publicaría como servidor web para uso público.

Como segundo objetivo, asumiendo que cumpliéramos el primero, se pensó en desarrollar una aplicación de escritorio en Java. Esta aplicación recibiría por consola el texto que se quiere resumir, haría una petición al servidor de Grafeno pasando por parámetro el texto, y recibiría por respuesta el texto resumido que se mostraría al usuario, también por consola.

### 4.2 Desarrollo

Para empezar con este nuevo desarrollo buscamos un entorno de trabajo más cómodo y práctico que una terminal de [Putty \[22\]](#). La herramienta [Eclipse \[23\]](#) cuenta con un IDE Python llamado [PyDev \[24\]](#) que permite desarrollar proyectos en este lenguaje. La pega es que algunas de las librerías que necesita Grafeno en Linux no son compatibles en Windows, por lo que descartamos esta opción.

Como alternativa a PyDev y para compensar las librerías, optamos por la herramienta [VirtualBox \[25\]](#) y ya que esta herramienta proporciona un entorno gráfico de Linux, se pensó en realizar todo el desarrollo en Eclipse, así no habría problemas con las librerías; además, a base de depuraciones sería más fácil llegar a comprender los pasos que sigue Grafeno. El inconveniente de esta opción fue que el desarrollo se haría en local y los avances en código no estarían compartidos en todo momento.



Finalmente nos decidimos por usar [WinSCP \[26\]](#) para el acceso a los ficheros y [Notepad++ \[27\]](#) para su lectura y edición. De ese modo todos los cambios se guardarán directamente en el repositorio y resultaba mucho más cómodo que trabajar desde la consola.

Para realizar los resúmenes Grafeno se sirve de FreeLing en la tarea de generar las dependencias semánticas de las frases. Además, que Grafeno estaba desarrollado y pensado para el inglés y tenía reglas para entender texto en este idioma, no en español. La cuestión era que FreeLing da nombres distintos a las dependencias en función del idioma con el que se trabaje. Por ejemplo, en inglés el sujeto se marca con "ncsubj", mientras que en español con "subj". Comenzamos entonces por identificar las diferencias entre las etiquetas usadas para estos dos idiomas. Una primera prueba de Grafeno, con texto en español (sin realizar ningún cambio por nuestra parte), nos hizo pensar que Grafeno ya estaba preparado para generar los resúmenes, pero únicamente daba resultados porque las operaciones que realiza Grafeno para generar resúmenes son independientes del idioma del texto, ya que se forman grafos por clustering en función de las palabras más referenciadas y se devuelven las frases de donde se producen las referencias más importantes.

Comenzamos el desarrollo buscando información sobre FreeLing. Como se señaló anteriormente, FreeLing ofrece sus servicios en varios idiomas. Dentro del directorio de instalación de FreeLing nos encontramos con el fichero de configuración que por defecto le corresponde al idioma español. Editamos este fichero de tal modo que se pareciera al que usaba Grafeno para obtener los resultados en inglés. Una vez fue estable, buscamos y encontramos el lugar donde Grafeno elige la configuración que se debe usar sobre FreeLing, y sustituimos la llamada al fichero de configuración de inglés por el del español. Se probó a ejecutar Grafeno para ver si los cambios eran suficientes y se seguía generando resúmenes. Para asegurar que la generación del resumen se debía a nuestros cambios hicimos pruebas directamente sobre FreeLing:

```
analyze -f grafeno/freeling_deps_es.cfg < texto.txt
```

Prueba1:

En esta primera prueba, dejamos los valores por defecto del fichero de configuración.

Dentro texto.txt tenemos:

Las naranjas son naranjas

Las el DA0FP0 0.988184  
naranjas naranja NCCP000 0.638706  
son ser VSIP3P0 0.995197  
naranjas naranja AQ0CP00 0.361294



## Prueba 2:

Con el mismo texto de entrada cambiamos en esta ocasión el formato de salida de Freeling para que esta vez nos muestre las dependencias semánticas de las palabras. Para ello cambiamos OutputLevel=dep y el formato de salida será tipo JSon OutputFormat=json. Obtenemos como resultado

```
{ "id": "1",
  "tokens" : [
    { "id" : "t1.1", "form" : "Las", "lemma" : "el",
      "tag" : "DA0FP0", "ctag" : "DA", "pos" : "determiner",
      "type" : "article", "gen" : "feminine", "num" : "plural"},
    { "id" : "t1.2", "form" : "naranjas", "lemma" : "naranja",
      "tag" : "NCCP000", "ctag" : "NC", "pos" : "noun",
      "type" : "common", "gen" : "common", "num" : "plural"},
    { "id" : "t1.3", "form" : "son", "lemma" : "ser",
      "tag" : "VSIP3P0", "ctag" : "VSI", "pos" : "verb",
      "type" : "semiauxiliary", "mood" : "indicative", "tense" : "present",
      "person" : "3", "num" : "plural"},
    { "id" : "t1.4", "form" : "naranjas", "lemma" : "naranja",
      "tag" : "AQ0CP00", "ctag" : "AQ", "pos" : "adjective",
      "type" : "qualificative", "gen" : "common", "num" : "plural"}],
  "constituents" : [
    { "label" : "grup-verb", "head" : "1", "children" : [
      { "label" : "sn", "children" : [
        { "label" : "espec-fp", "children" : [
          { "label" : "j-fp", "head" : "1", "children" : [
            { "leaf" : "1", "head" : "1", "token" : "t1.1", "word" : "Las"}
          ]
        }
      ]
    },
    { "label" : "grup-nom-fp", "head" : "1", "children" : [
      { "label" : "n-fp", "head" : "1", "children" : [
        { "leaf" : "1", "head" : "1", "token" : "t1.2", "word" : "naranjas"}
      ]
    }
  ]
},
    { "label" : "verb", "head" : "1", "children" : [
      { "leaf" : "1", "head" : "1", "token" : "t1.3", "word" : "son"}
    ]
},
    { "label" : "s-adj", "children" : [
      { "label" : "s-a-mp", "head" : "1", "children" : [
        { "label" : "a-mp", "head" : "1", "children" : [
          { "leaf" : "1", "head" : "1", "token" : "t1.4", "word" : "naranjas"}
        ]
      }
    ]
  ]
},
  ],
  "dependencies" : [
    { "token" : "t1.3", "function" : "top", "word" : "son", "children" : [
      { "token" : "t1.2", "function" : "subj", "word" : "naranjas",
        "children" : [
          { "token" : "t1.1", "function" : "spec", "word" : "Las"}
        ]
      },
      { "token" : "t1.4", "function" : "attr", "word" : "naranjas"}
    ]
  ]
}]}
```

De esta salida apreciamos que detecta correctamente las palabras y que el análisis que se hace sobre ellas es en español:

```
{ "id" : "t1.1", "form" : "Las", "lemma" : "el", "tag" : "DA0FP0", "ctag" : "DA", "pos" :
"determiner", "type" : "article", "gen" : "feminine", "num" : "plural"}
```

Lo cual quiere decir que “Las” es un determinante artículo femenino plural.

Como las pruebas sobre Freeling daban resultados correctos, se asumió que la configuración

y adaptación al español estaba completa. Se publicó esta nueva versión de Grafeno como servidor web para realizar los resúmenes en español y se procedió a desarrollar la aplicación de escritorio.

Para el desarrollo de la aplicación de escritorio usamos Eclipse. Esta aplicación consistía en una consola en la cual el usuario debía introducir el texto a resumir. Una vez leído, se haría la petición al servidor de Grafeno, se extraería el texto resumido de la respuesta, y finalmente se mostraría este al usuario, también por consola.

## 4.3 Resultados y discusión crítica

A diferencia del primer objetivo (la fusión de proyectos) tuvimos mejores resultados en esta ocasión. A partir de un documento de texto en español generamos un texto resumido del documento original. En el resumen generado se puede apreciar un buen uso de la gramática y la ortografía, y el resultado parece un resumen redactado por una persona puesto que es precisamente así. El contenido del resumen son extractos esenciales del documento original y representan lo más importante del mismo.

Dado que el objetivo de esta herramienta es facilitar la comprensión del texto, dividimos el texto en partes y párrafos para facilitar su visualización, ya que cuando el texto es demasiado espeso afecta a la visión y dificulta la lectura.

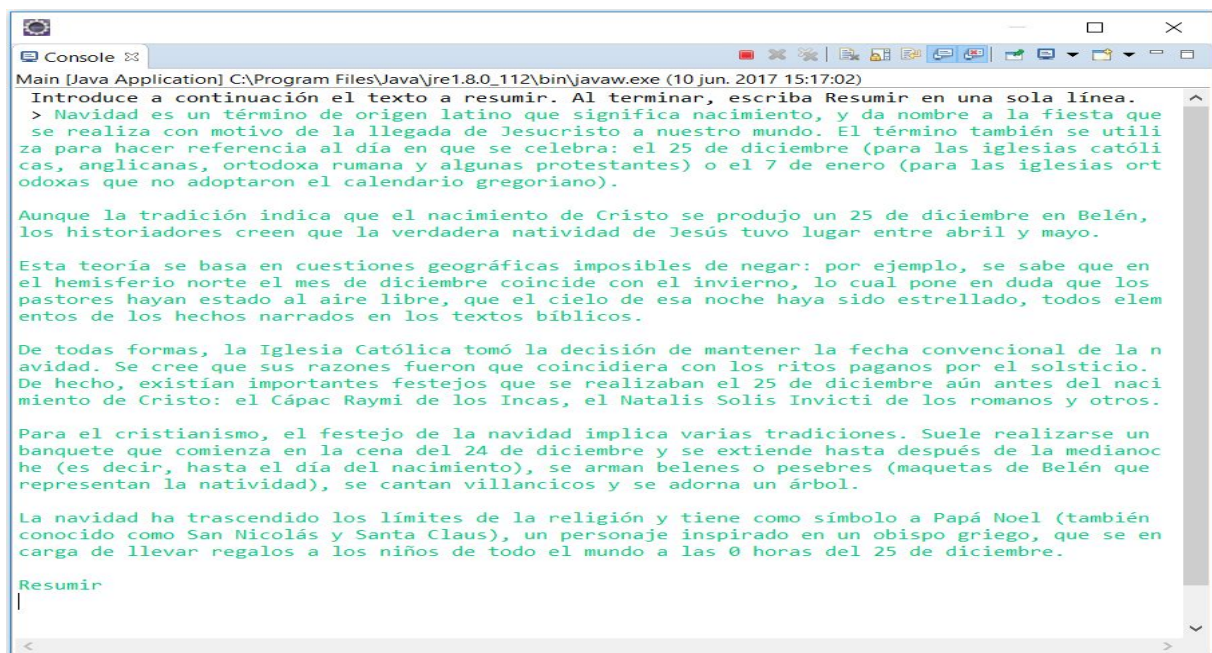
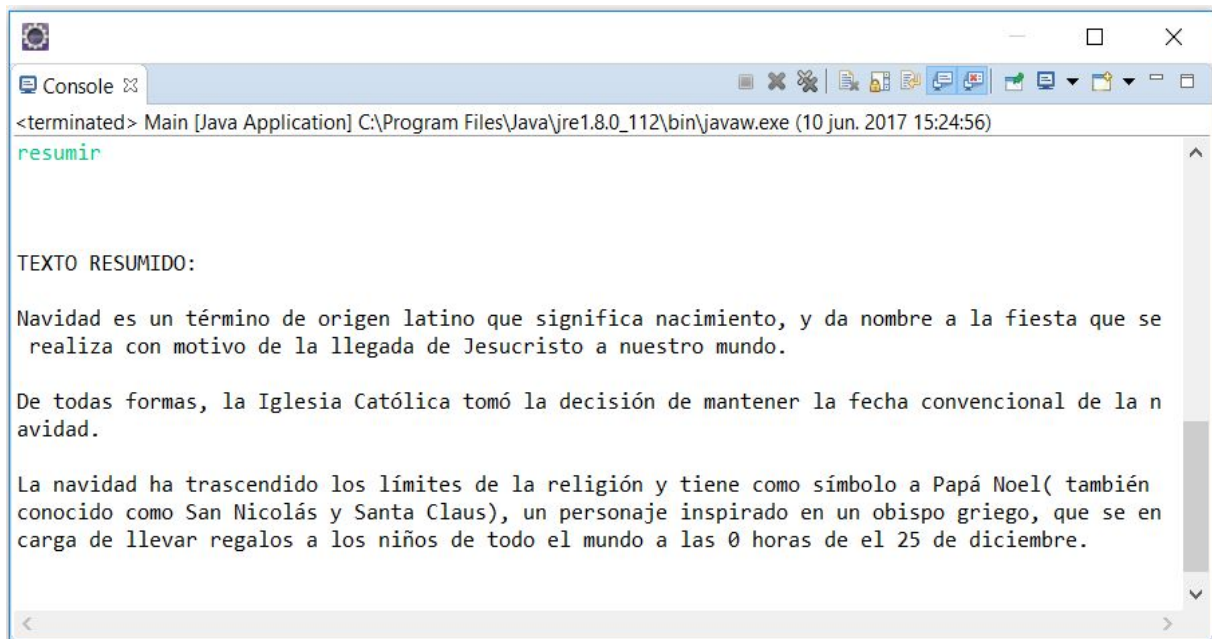


Figura 12: Introducción de texto desde la consola de Eclipse

El resumen generado consiste en un resumen por extracción, es decir, a partir del texto original se extraen las frases más importantes relacionadas con el tema principal. El problema de este proceso es que la primera frase extraída puede comenzar con un conector del tipo “por ejemplo”, “además”, etc. Es decir, el texto resumido podría no comenzar

hablando sobre el tema, lo cual puede dejar al lector desorientado. Para poder corregir esto puede usarse la técnica de generación simple de texto del proyecto de máster: la generación de texto comienza colocando el concepto sujeto.



```
<terminated> Main [Java Application] C:\Program Files\Java\jre1.8.0_112\bin\javaw.exe (10 jun. 2017 15:24:56)
resumir

TEXTO RESUMIDO:

Navidad es un término de origen latino que significa nacimiento, y da nombre a la fiesta que se realiza con motivo de la llegada de Jesucristo a nuestro mundo.

De todas formas, la Iglesia Católica tomó la decisión de mantener la fecha convencional de la Navidad.

La Navidad ha trascendido los límites de la religión y tiene como símbolo a Papá Noel( también conocido como San Nicolás y Santa Claus), un personaje inspirado en un obispo griego, que se encarga de llevar regalos a los niños de todo el mundo a las 0 horas de el 25 de diciembre.
```

*Figura 13: Texto resumido, también por consola de Eclipse.*

*El segundo párrafo hace referencia a las fechas de la Navidad, pero la frase comienza con un conector.*

Por último, dado que la longitud de cada texto puede variar, pedimos que se implementara en el servidor web una funcionalidad que permitiera al usuario especificar la longitud del resumen, ya que, en algunos casos, con la longitud por defecto, el resumen resultante era demasiado corto.

En un primer momento nuestro trabajo de Grafeno siempre ha sido una rama o branch del proyecto Grafeno desarrollado por Antonio F. G. Sevilla y colaboradores. Al tratarse de un branch que ya ha sido completado y que ha llegado a un estado favorable, procedimos a actualizar y publicar los cambios en [este repositorio de GitHub](#) como un fork del proyecto principal.

## Capítulo 5

# Simplificación

En este capítulo nos centramos en el módulo encargado de hacer del resumen un texto aún más sencillo de entender. Nos centraremos en la fase de simplificación.

El objetivo de esta fase era mejorar el resumen generado de tal modo que resultase más sencillo de entender que el resumen original. Desarrollamos un módulo extra capaz de simplificar el texto resumido, convirtiendo las oraciones complejas en oraciones más simples. Para lograrlo, se extrae del texto información innecesaria, y el vocabulario complejo se reemplaza por uno más sencillo basándonos en el criterio de [unos servidores web de accesibilidad](#), trabajos de fin de grado anteriores a este. Todo esto se haría manteniendo la información y el significado original del texto.

Por otro lado, ya que el usuario puede no estar familiarizado con el uso de la consola y cómo ejecutar la herramienta desde allí, cambiaremos el diseño de la aplicación para que se pueda usar con una interfaz gráfica. Daremos al usuario tanto la posibilidad de seleccionar el fichero de texto a resumir, como la opción de redactar el texto directamente desde la interfaz. Finalmente, y con un solo clic se procesará el texto y se mostrarán, en vistas independientes, el texto original, el texto resumido y el texto de la simplificación. Al tratarse de una herramienta de apoyo, todos los textos podrán ser editados una vez generado el resumen, para que el usuario pueda personalizarlos a su gusto.

### 5.1 Desarrollo

Para llevar a cabo esta etapa empezamos desarrollando la interfaz. Puesto que el procesamiento interno ya estaba completado bastaba con adaptar el código cambiando el lugar de las llamadas.

La primera versión consistía en una vista con tres botones de menú (abrir fichero, resumir y limpiar texto) y un cuadro de texto donde introducir el texto. Básicamente al clicar en el botón de resumen enviaba el texto de entrada al servidor de Grafeno, y la respuesta de éste se mostraba en una ventana nueva, tal y como se muestra en la Figura 14.

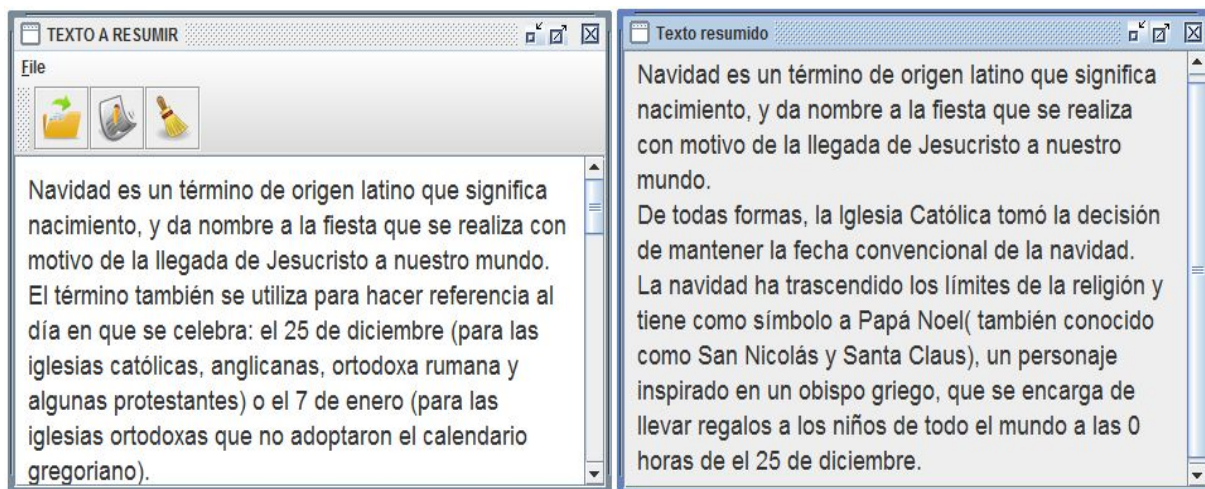


Figura 14: Primera versión, cuadro de texto y texto resumido

En la segunda versión la ventana de resumen estaba acoplada a la ventana de principal, de modo que resultara más fácil hacer las comparaciones entre textos. Además, dado que queríamos que esta herramienta sirviera de apoyo para generar resúmenes, dimos al usuario la posibilidad de editar tanto el texto original como el texto resumido directamente desde los cuadros de texto. Por último, agregamos un botón de guardado que permitía guardar el texto resumido una vez terminada la edición, tal y como se muestra en la Figura 15.

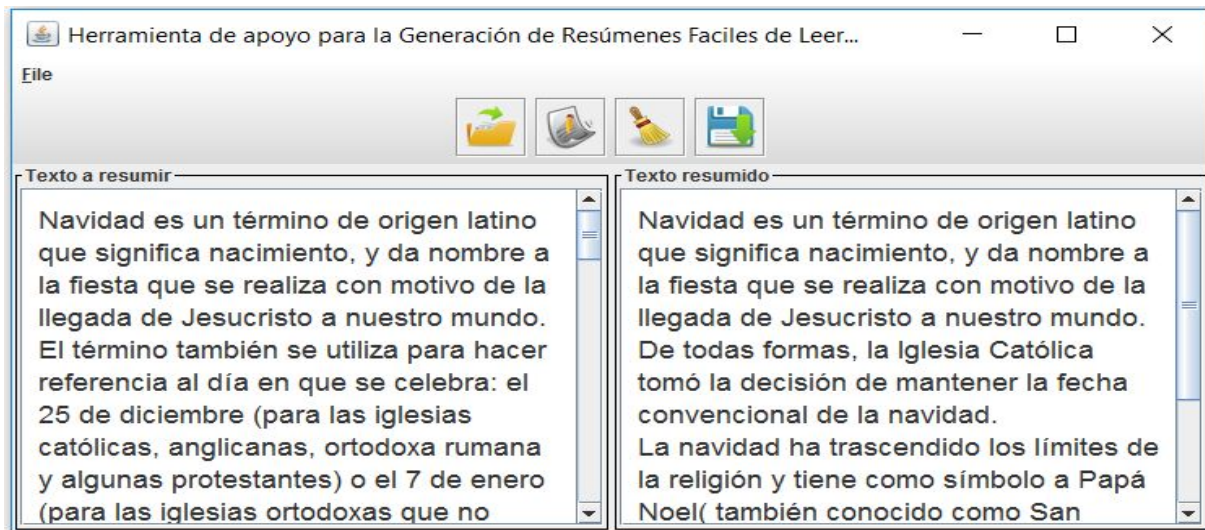


Figura 15: Segunda versión, cuadro de texto y texto resumido unificados. Opción de guardar

En una tercera versión de la aplicación incluimos una tercera vista para contener el resumen simplificado. Aquí comenzó la parte más importante de esta etapa.

Para el resumen simplificado analizamos el texto resumido que nos devuelve el servidor de Grafeno. Procesamos el texto, y retiramos de él información que consideramos innecesaria, (como el texto explicativo que se escribe entre paréntesis). Una vez limpiamos el texto



resumido buscamos palabra por palabra aquellas que se considerasen “difíciles” y las sustituimos por un sinónimo más “fácil” de entender. Para saber si una palabra era considerada difícil o no, usamos dos de los servicios web de accesibilidad. La diferencia entre uno y otro es el formato de salida de los datos (JSON o XML). Implementamos dos funciones, una para cada servicio, para que en caso de que falle la primera se llame a la segunda, y en caso de que ésta también falle, dejamos la palabra original:

- <http://sesat.fdi.ucm.es:8080/servicios/rest/palabras/json/palabra>
- <http://sesat.fdi.ucm.es:8080/servicios/rest/palabras/xml/palabra>

Una de las partes más complicadas de esta etapa fue la sustitución de las palabras difíciles por los sinónimos adecuados. Para obtener los sinónimos también usamos dos recursos:

- <http://sesat.fdi.ucm.es:8080/servicios/rest/sinonimos/json/palabra>
- <http://sesat.fdi.ucm.es:8080/servicios/rest/conversion/json/palabra>

El primer recurso nos retornaba, para una palabra dada, un listado de sus sinónimos; el segundo recurso, para una palabra considerada difícil, devolvía su sinónimo fácil. Como el segundo recurso se acercaba más a nuestros objetivos, utilizamos éste.

En una cuarta versión, para orientar al usuario y marcar las diferencias entre el texto resumido y el texto simplificado señalamos en la vista del resumen simplificado aquellas palabras reemplazadas por sinónimos. Además, retiramos del resumen simplificado aquella información innecesaria y que solo sirve para extender el texto. Quitamos el texto contenido entre paréntesis y separamos cada párrafo con un salto de línea para tener una visión más clara y legible de la información, tal y como se muestra en la Figura 16.

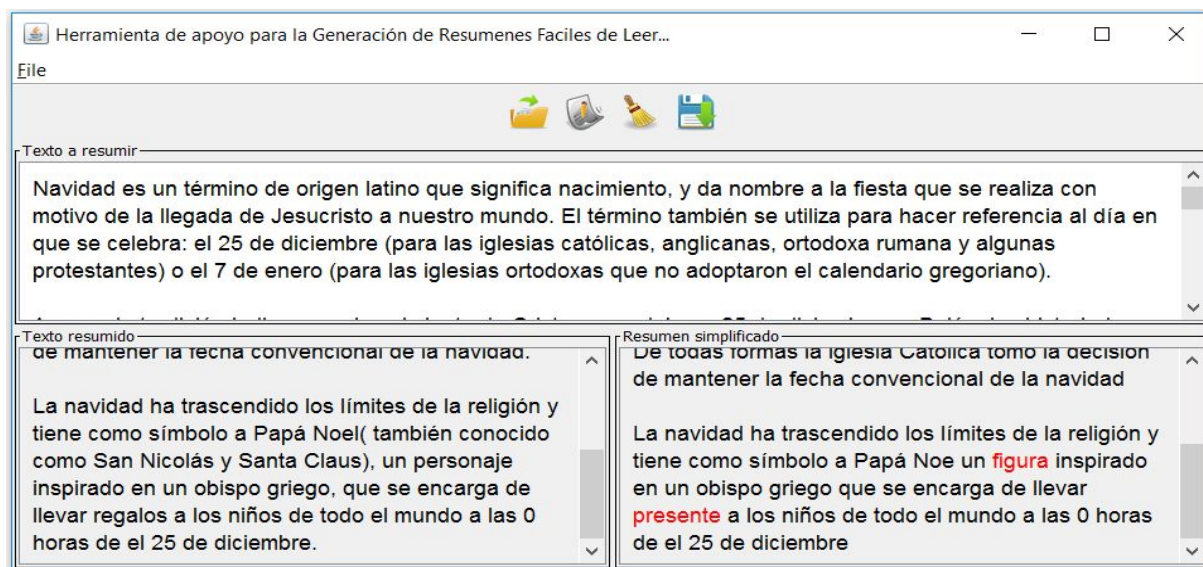


Figura 16: Cuarta versión, texto simplificado

Uno de los problemas que tuvo esta cuarta versión fue que para comparar las palabras cambiadas entre el texto resumido y el resumen simplificado creamos dos arrays de palabras de ambos textos y las comparábamos: si eran iguales se escribían en un color, y si eran

distintas en otro. Dado que los arrays eran únicamente de palabras, en el momento de escribirlas se perdía la posición donde estaban situados los puntos, comas, y demás signos de puntuación. Como resultado, el texto simplificado se convertía en una secuencia de palabras y era difícil saber dónde empezaba y dónde terminaba una frase, luego más que simplificar el resumen hacíamos que fuera más difícil de entender.

En la quinta versión reestructuramos el proyecto y encontramos el modo de corregir el problema del resumen simplificado de la versión anterior. En la versión anterior se comparaban las palabras entre los arrays y se escribían directamente en la vista del resumen simplificado. En esta versión se comparan también las palabras desde los arrays, pero esta vez cuando se encontraban dos que no coincidían se extraía del texto original la subcadena que iba desde el inicio hasta la posición de la palabra a comparar, y esta subcadena se concatenaba con la palabra cambiada. Dado que ahora el resumen simplificado se generaba a partir del texto resumido y solo cambiamos del primero las palabras que no coincidían en los arrays, podíamos conservar los signos de puntuación y el resumen simplificado se ganaba su nombre, tal y como se muestra en la Figura 17.

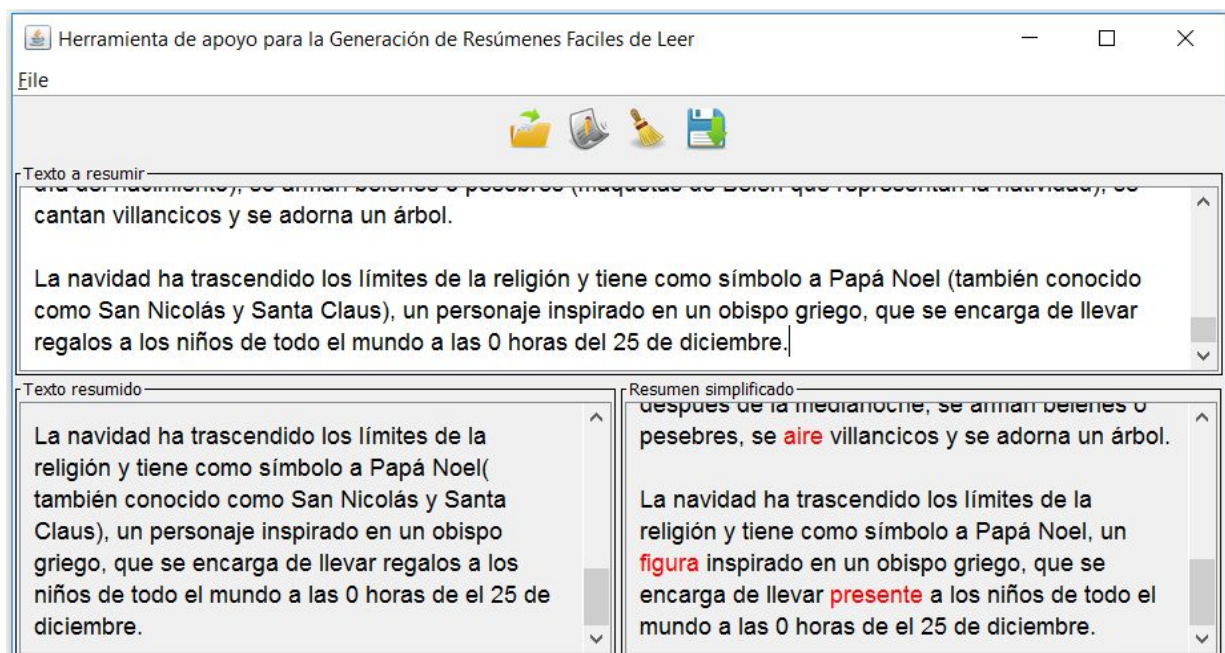


Figura 17: Quinta versión, recuperación de signos de puntuación

Esta quinta versión arrastraba otro problema de la versión anterior, en la que: a la hora de sustituir una palabra difícil por su sinónimo, la palabra sustituta podía perder no solo la dependencia de género y número con la palabra a sustituir, sino también podría tener una categoría gramatical distinta.

En una sexta versión intentamos corregir el problema de la concordancia entre **género** y **número**. Dado que el sinónimo devuelto por el recurso de simplificación no depende de nosotros, para solventar este problema corregimos las dependencias cambiando la palabra anterior, generalmente un determinante, por su equivalente que concordara en

género y número con la palabra sustituta.

En el entorno de trabajo de Grafeno contamos con el comando analyze de FreeLing. Pedimos que se implementara en el servidor web de Grafeno una funcionalidad que devolviera, para un conjunto de palabras, el género y número de cada una, siempre y cuando lo tuvieran.

Una vez finalizada la nueva funcionalidad de Grafeno, desarrollamos otro módulo independiente que, a partir de un determinante cualquiera, devolviera su equivalente según el género y número que se indicara. Así, si tenemos por ejemplo la frase **“tengo las pinturas en la mesa”** la palabra **“pinturas”** se considera difícil y se sustituye por **“cuadro”**. En la versión anterior, el texto simplificado sería **“tengo las cuadro en la mesa”**, pero en esta versión el resultado que obtenemos es **“tengo el cuadro en la mesa”**. De este modo logramos subsanar que los sinónimos que recibimos pierdan la relación de género y número con la palabra original, y logramos que el resultado sea más apropiado, tal y como se muestra en la Figura 18.

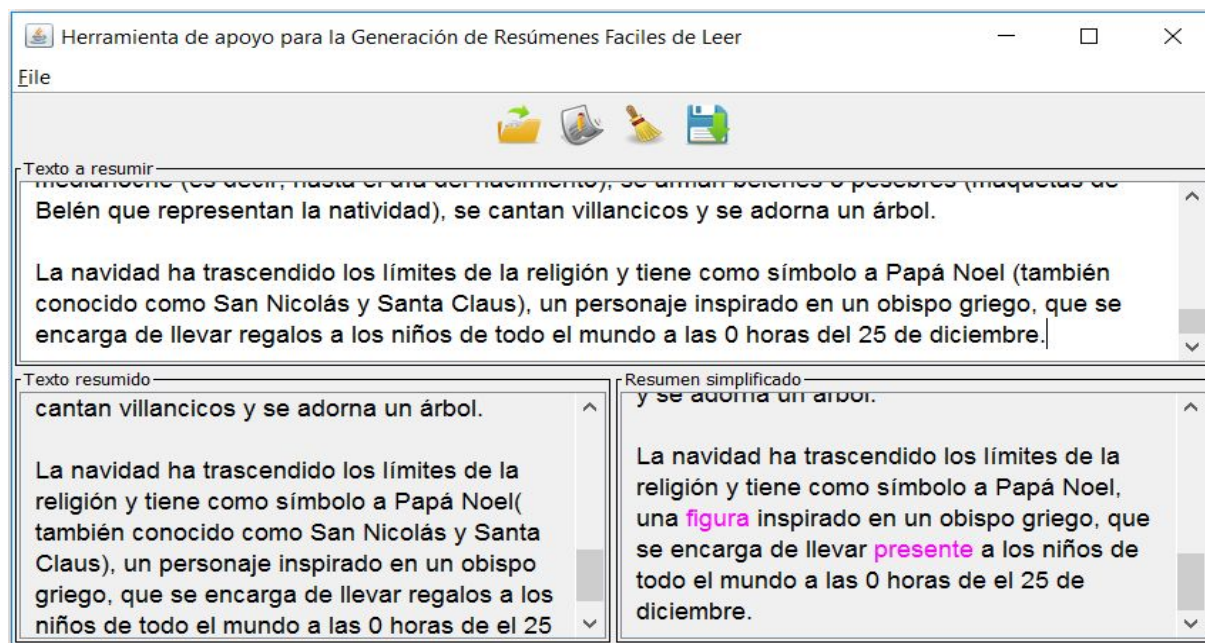


Figura 18: Sexta versión, concordancia de género y número.

Se marcan en morado las palabras que han pasado por el tratamiento de género y número.

## 5.2 Resultados y discusión crítica

En esta fase pudimos obtener buenos resultados y cumplimos con los objetivos que nos propusimos. No obstante, la versión final tiene muchos aspectos que mejorar. Contamos en el Estado del Arte que el proyecto Simplext proponía dos tareas de simplificación: la sintáctica y la semántica. Haremos una comparación entre lo que propone y la funcionalidad que ofrecemos.



Para la simplificación semántica hacemos la sustitución de palabras complicadas por otras más fáciles de entender. Sin embargo, con independencia de qué recurso utilicemos para la sustitución de las palabras difíciles, la dificultad con la que nos encontramos es que el sinónimo que nos ofrece el recurso puede no tener concordancia en género y número con la palabra sustituida, o lo que es peor, el sinónimo devuelto podría diferir en categoría con la palabra sustituida dando, por ejemplo, un verbo por un adjetivo. El primer problema lo resolvimos parcialmente manteniendo las dependencias con determinantes. No obstante, no pudimos resolver los casos de sustitución por verbos o cuando la palabra sustituta dependía en género y número con la siguiente a ésta, tal y como se muestra en la Figura 19.

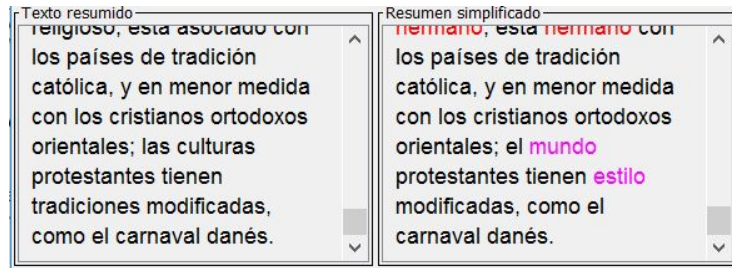


Figura 19: Simplificación semántica, dependencias gramaticales

En este ejemplo, la palabra “**protestantes**” depende en género y número con “**culturas**” que se sustituye por “**mundo**”. El resultado que obtenemos es “**el mundo protestantes**”, en lugar de “**los mundos protestantes**” o “**el mundo protestante**”.

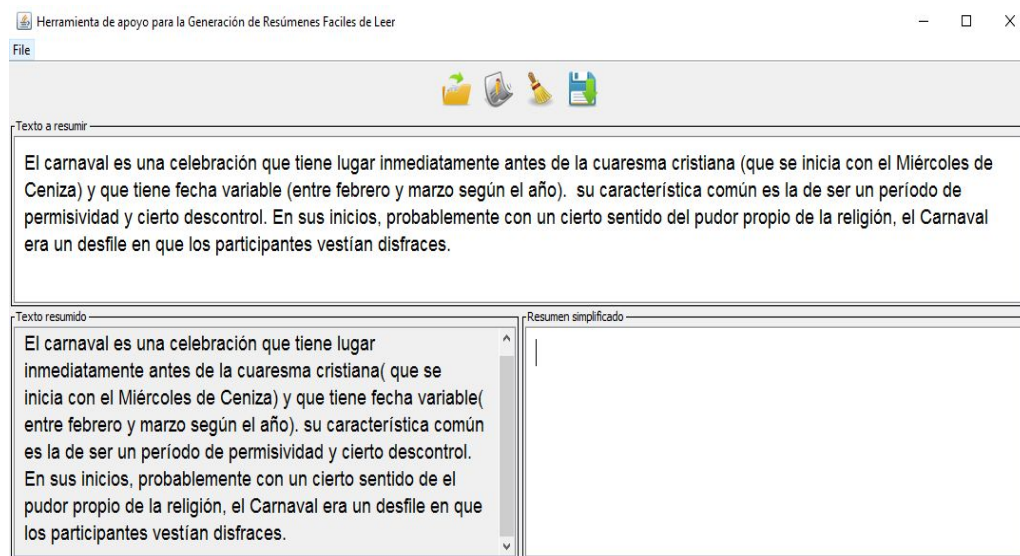
## Capítulo 6:

# Evaluación de la Aplicación

## 6.1 Comparación de HAGRes con Simplext

Con el objetivo de comprobar la fiabilidad de la información en la aplicación, se ha llevado a cabo una evaluación de resultados entre la herramienta desarrollada en este proyecto **HAGRes** y una demo de la herramienta **Simplext**.

La Figura 20 muestra el resumen por extracción del texto analizado en **HAGRes**.



*Figura 20: Generación del Resumen.*

Una vez obtenido el resumen, realizamos la simplificación del texto, tal y como se muestra en la Figura 21:

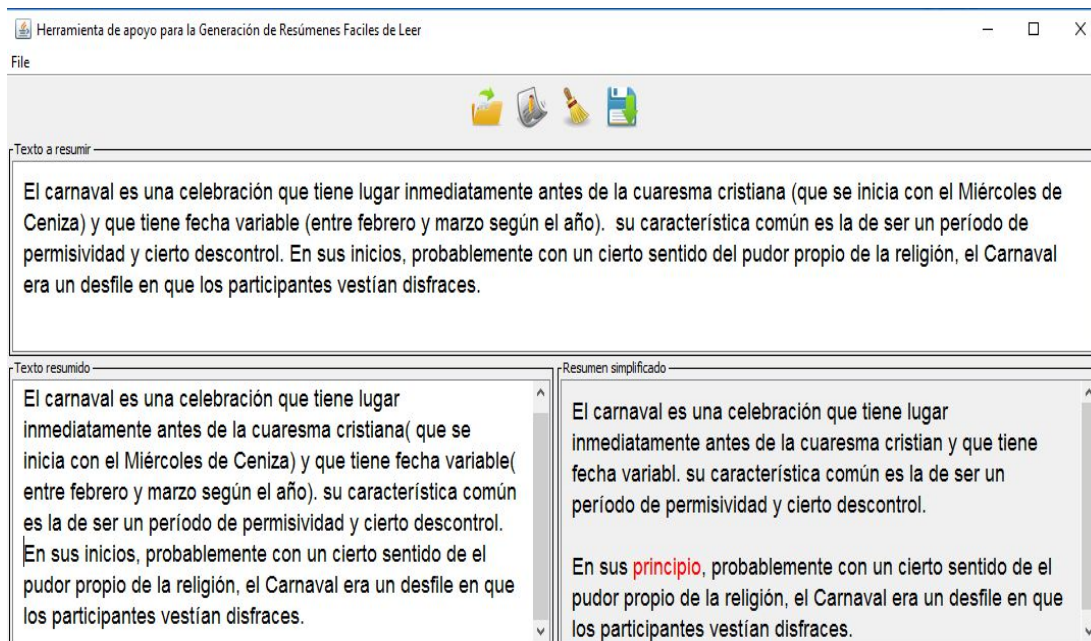


Figura 21: Simplificación de texto en **HAGRes**.

La Figura 22 muestra la simplificación del resumen usado anteriormente, pero en **Simplext**.



Figura 22: Simplificación de texto en **Simplext**.

## 6.2.Resultados de la evaluación usando HAGRes y Simplext

Realizando la comparación entre ambas aplicaciones, podemos ver la similitud en el resultado obtenido, aunque con pequeños matices que bien podrían ser objeto de mejoras futuras. En el caso de HAGRes, al encontrar en el servicio web la Palabra **inicios** como difícil, está realiza la búsqueda de un sinónimo de dicha palabra y realiza el cambio, en el caso de Simplext, este cambio no se lleva a cabo.

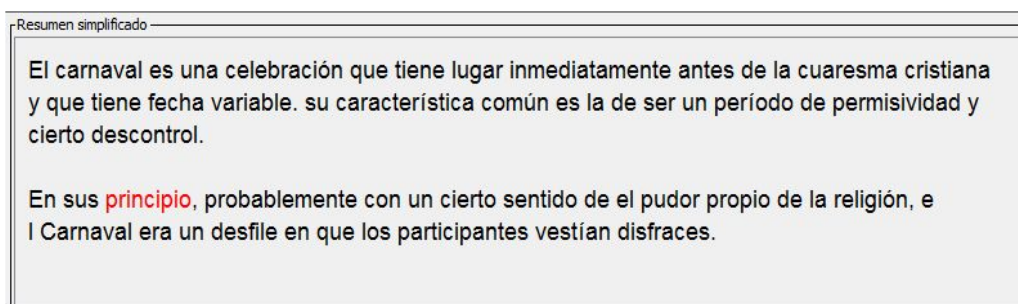


Figura 23: Análisis de la Simplificación en **HAGRes**.

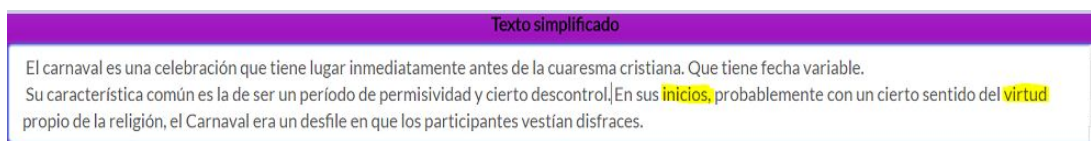


Figura 24: Análisis de la Simplificación en **Simplext**.

## Capítulo 7:

# Conclusiones y trabajos futuros

### 7.1 Conclusiones

Una vez finalizado el desarrollo del proyecto, se pueden establecer las siguientes conclusiones:

- Como consecuencia de la sociedad de la información en la que vivimos se hace necesario satisfacer una demanda creciente relativa al acceso a los textos a través de una lectura fácil.
- Grafeno, ha permitido llevar a cabo la funcionalidad requerida para la generación del resumen de un texto en español, donde el contenido del resumen es lo más importante del documento original, por lo que podemos concluir que se ha cumplido con el objetivo establecido, siendo Grafeno una herramienta muy flexible y de gran utilidad para el desarrollo de este proyecto
- Para realizar la simplificación de un texto, ya sea cualquier texto de entrada o el texto del resumen generado por Grafeno, este nos devuelve una versión simplificada del mismo, sustituyendo las palabras más complejas por otras más fáciles de comprender. Aunque con pequeños matices que bien podrían ser objeto de trabajos futuros, podríamos decir que se ha cumplido con el objetivo propuesto.
- Ambas funcionalidades se han integrado en una aplicación de escritorio, que ha resultado óptima dados los resultados obtenidos. Esta herramienta se desarrolló en un entorno Java y lleva por nombre **HAGRes** (Herramienta de Apoyo para la Generación de Resúmenes Fáciles de Leer en español).

## 7.2 Trabajos futuros

Durante el desarrollo de este proyecto se han ido planteando mejoras que no se han llevado a cabo. A continuación, se proponen características que podrían mejorar este proyecto.

Para la simplificación semántica hacemos la sustitución de palabras complicadas por otras más fáciles de entender, sin embargo, la dificultad con la que nos encontramos es que independientemente del recurso que utilicemos para la sustitución de estas palabras, el sinónimo que nos ofrece el recurso puede no tener concordancia en género y número, este problema se logró solventar parcialmente manteniendo las dependencias con determinantes, no obstante, no pudimos resolver son los casos de sustitución por verbos, donde el sinónimo devuelto podría diferir en categoría con la palabra sustituida dando, por ejemplo, un verbo por un adjetivo

Un posible trabajo futuro podría consistir en utilizar recursos más eficaces para la sustitución, o desarrollar otros módulos capaces de, por ejemplo:

- conservar la conjugación de los verbos cuando se sustituya uno por otro
- aplicar reglas de la gramática para convertir las palabras a su forma plural cuando fuera necesario, para mantener la concordancia
- dotar a la aplicación de “memoria” e indicar qué palabras, bien por ser ambiguas, bien por otro motivo, no deben sustituirse en la simplificación

Para la Simplificación sintáctica, la cual se basa en quitar los textos entre paréntesis y en la que se ignora por completo los conectores. Esto podría mejorarse aprovechando las dependencias semánticas que ofrece FreeLing para convertir oraciones compuestas en oraciones simples. Una opción sería sustituir los pronombres de la subordinada por los sustantivos a los que hacen referencia en la oración principal. También sería de mucha utilidad dotar a las frases del orden canónico de sujeto-verbo-predicado. Para ello se podría utilizar el grafo de dependencias semánticas de FreeLing y según los valores de cada nodo detectar aquellas palabras que pertenecen al sujeto, y las que pertenecen al predicado, dejando el verbo entre medias.

## Capítulo 8

# Bibliografía y Referencias

- [1] <http://www.the-wow-collection.com/software/NLPforBeginners.pdf>
- [2] <http://eprints.ucm.es/39327/>
- [3] <http://eprints.ucm.es/38686/>
- [4] <http://nil.fdi.ucm.es/?q=node/642>
- [5] <https://github.com/agarsev/grafeno>
- [6] <http://www-ai.ijs.si/eliza/eliza.html>
- [7] <http://eurotra.eu/>
- [8] <http://journal.sepln.org/sepln/ojs/ojs/index.php/pln>
- [9] <http://w3c.es/Divulgacion/GuiasBreves/Accesibilidad>
- [10] <http://accesibilidadweb.dlsi.ua.es/?menu=definicion>
- [11] <https://www.ifla.org/files/assets/hq/publications/professional-report/120-es.pdf>
- [12] <http://www.plenainclusion.org/sites/default/files/lectura-facil-metodos.pdf>
- [13] <https://www.w3.org/TR/WCAG10/>
- [14] <http://simplext.taln.upf.edu/>
- [16] <http://www.noticiasfacil.es/ES/Paginas/index.aspx>
- [16] <http://www.discapnet.es/Castellano/Paginas/default.aspx>
- [17] <http://www.talp.upc.edu/>
- [18] <http://nlp.lsi.upc.edu/freeling/demo/demo.php>
- [19] <http://wordnet.princeton.edu/>
- [20] <http://conceptnet-api-1.media.mit.edu/>
- [21] <http://asm.ow2.org/asm50/javadoc/user/org/objectweb/asm/util/Textifier.html>
- [22] <http://www.putty.org/>
- [23] <https://eclipse.org/>
- [24] <http://www.pydev.org/>
- [25] <https://www.virtualbox.org/>
- [26] <https://winscp.net/eng/docs/lang:es>
- [27] <https://notepad-plus-plus.org/>

Putty + Xming:

- <http://www.ajpdsoft.com/modules.php?name=News&file=article&sid=384>
- <https://microteknologias.wordpress.com/2008/04/14/xmingputty-administracion-grafica-y-remota-de-un-servidor-linux/>

## Apéndice 1

# Manual de usuario

### Ap 1.1 Interfaz gráfica

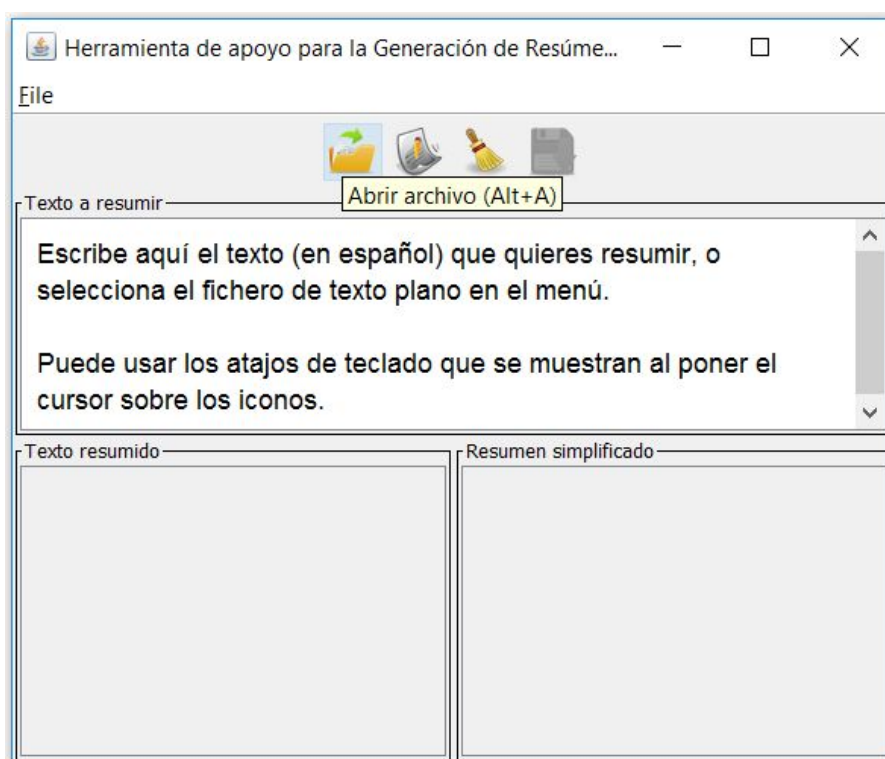


Figura 25: Vista de inicio

Cuando se inicia la aplicación se presenta esta ventana. En el cuadro superior llamado *Texto a resumir* se puede introducir manualmente el texto que va a ser objeto de resumen o se puede hacer *clic* en el primer icono (comando Alt+A) para cargar un fichero con el texto que se quiere resumir.

En el caso de cargar un fichero se abre una ventana nueva denominada “**Abrir texto a resumir**” que permite elegir el fichero de texto plano con el texto que se quiera trabajar.



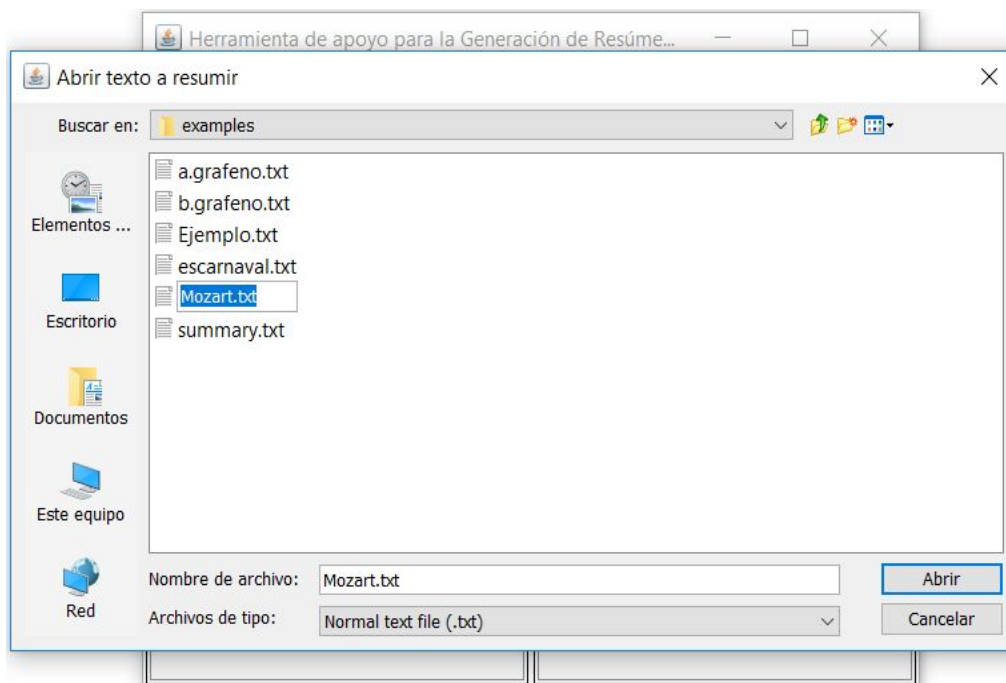


Figura 26: Cargar fichero de texto

Una vez elegido el fichero, se escribe su contenido en el cuadro del texto a resumir. En el caso de que falte o sobre contenido, se puede editar directamente desde la interfaz.

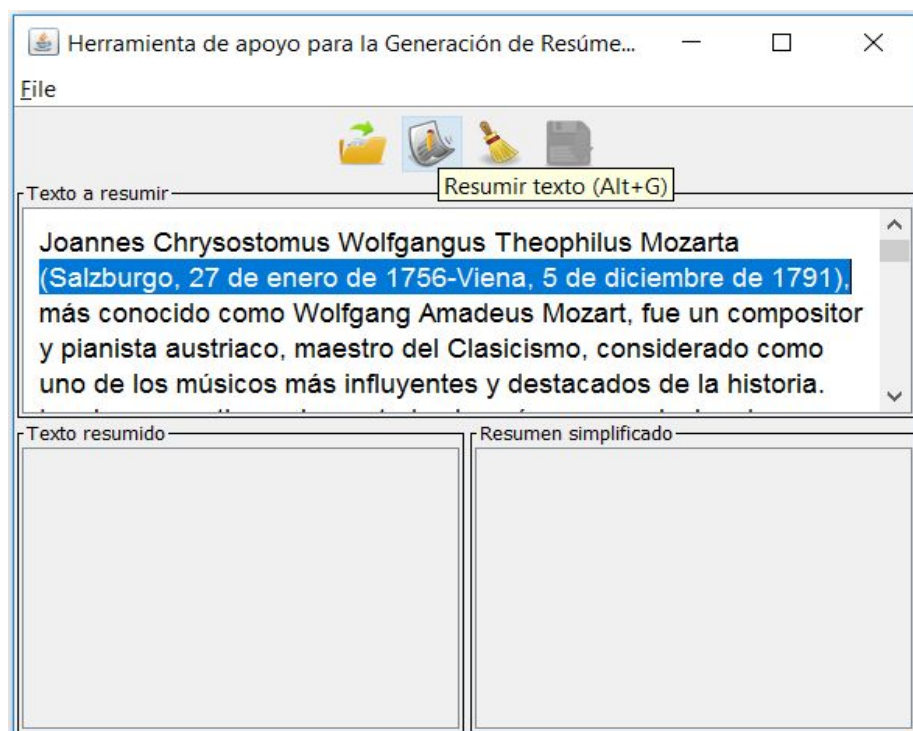


Figura 27: Editar texto a resumir, texto de entrada

Una vez esté preparado, se puede pasar al segundo icono denominado *Resumir texto* (Comando Alt+G). Cuando se resume el texto se cargan las dos vistas inferiores: Texto resumido y Resumen simplificado. En la primera está el resumen original proporcionado por el servidor de Grafeno; en el segundo está el resumen simplificado, es decir el resumen cuyas palabras difíciles se han cambiado por otros sinónimos más fáciles de entender:

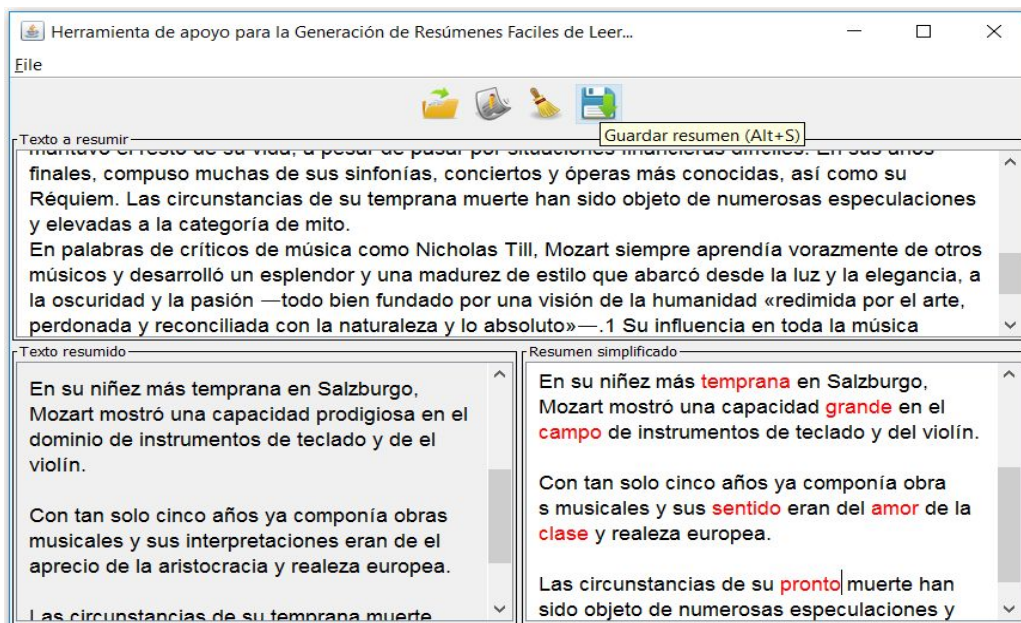


Figura 28: Resumen de texto

En color rojo se marcan aquellas palabras que han sido sustituidas. Si se diera el caso de que el cambio no fuera correcto, y tratándose ésta de una herramienta de apoyo, tanto la vista del texto resumido como la del resumen simplificado son editables, de esta forma se puede personalizar el resultado final obtenido.

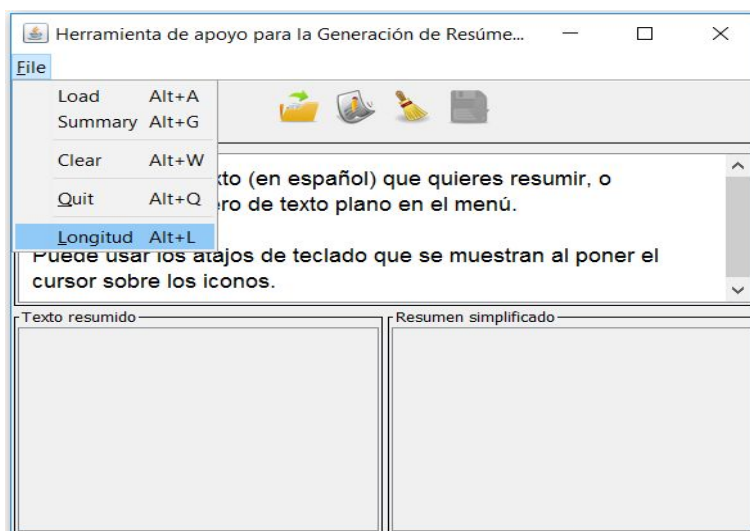


Figura 29: Cambiar longitud del resumen

Puede ocurrir que la extensión del resumen sea demasiado corta, para esta situación se ha agregado la opción de cambiar la longitud (número aproximado de palabras) del resumen. Para ello simplemente accedemos a la opción *Longitud* (comando Alt+L) disponible en File, desde la barra de herramientas.

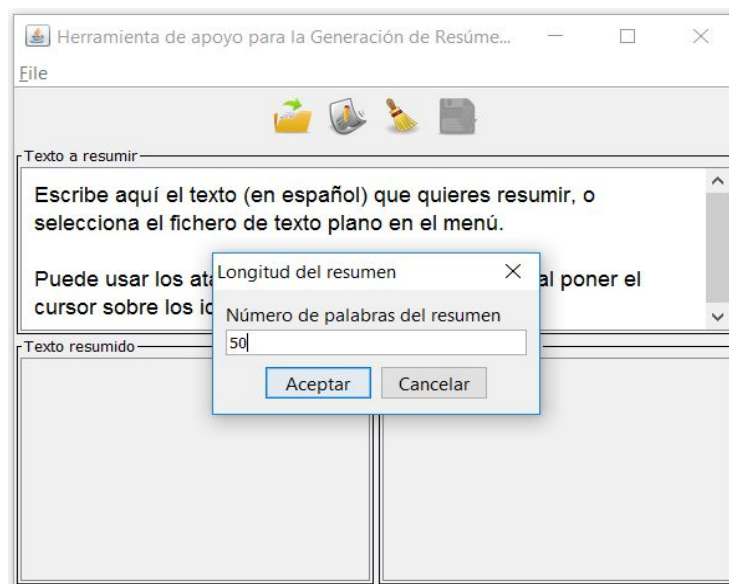


Figura 30: Especificar longitud del resumen

Cuando la longitud solicitada es menor que lo mínimo que se puede ofrecer, se ignora la petición del usuario y se devuelve el resumen con su longitud por defecto.

Una vez el resumen está completo, se activa el cuarto icono, Guardar resumen (Comando Alt+S).

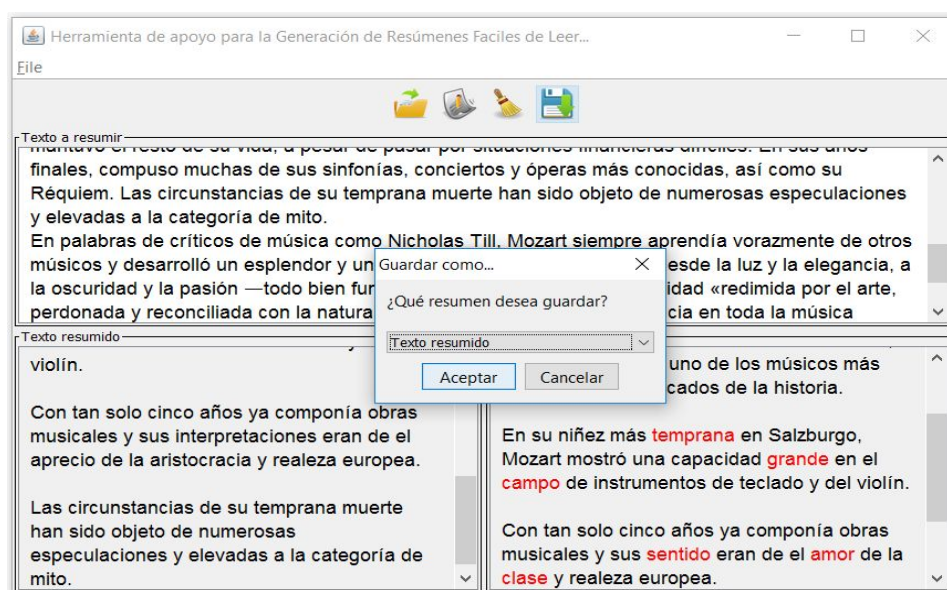


Figura 31: Guardar resumen (1)

Ya que tanto el texto resumido como el texto simplificado son editables, la opción de guardar permite, o bien guardar el resumen, o bien la simplificación, o bien guardar ambos:

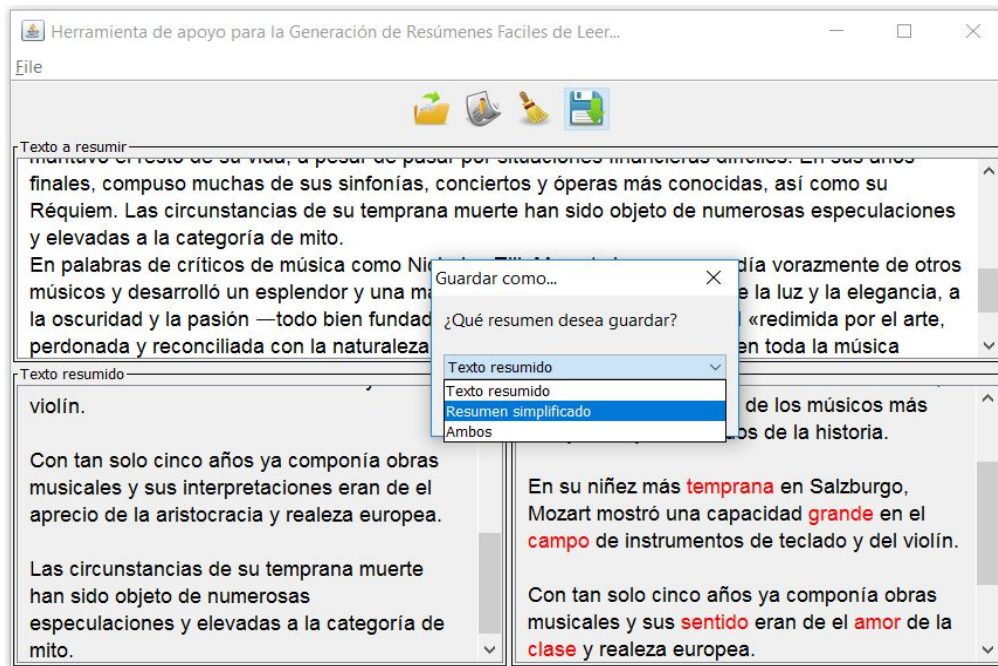


Figura 32: Guardar Resumen (2)

Una vez se acepte la opción deseada, se abre una ventana para elegir el directorio y el nombre con el que se quiere guardar el texto. Por defecto, todos los documentos se guardan con la extensión \*.hagres.txt.



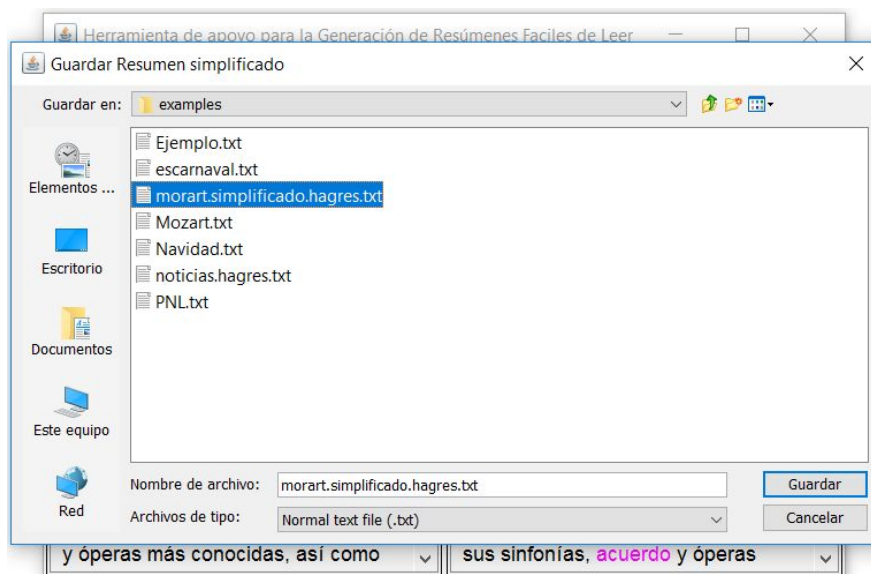


Figura 33: Guardar Resumen (3)

Una vez se termina de editar el texto, se puede pasar a otro sin necesidad de reiniciar la aplicación, basta con editar el texto de entrada, cambiar de fichero o limpiar las vistas con el tercer icono *Limpiar texto* (comando Alt+W):

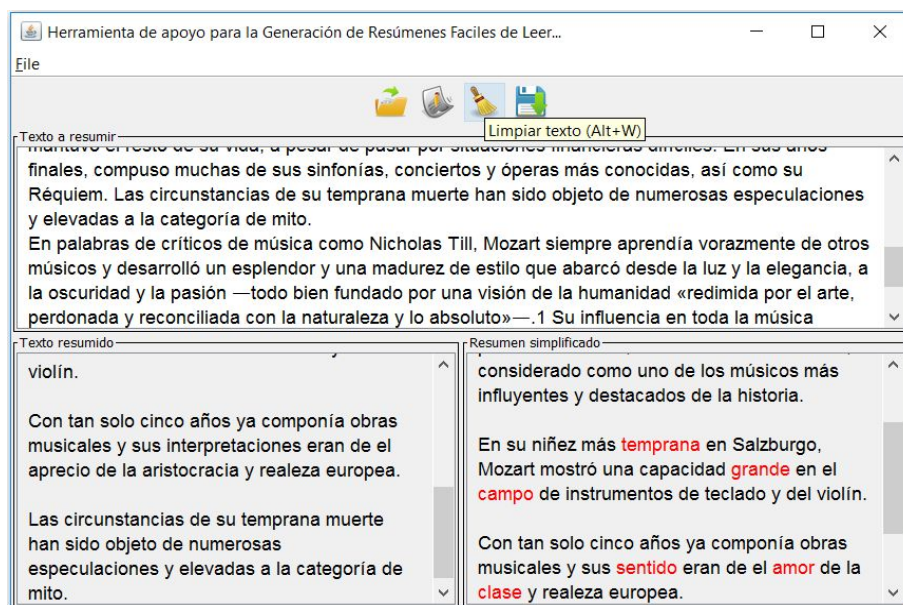


Figura 34: Reiniciar, limpiar vista.

Al limpiar el texto, se vuelve a la vista de inicio y se bloquea nuevamente el botón de guardado hasta que vuelva a generarse otro resumen.

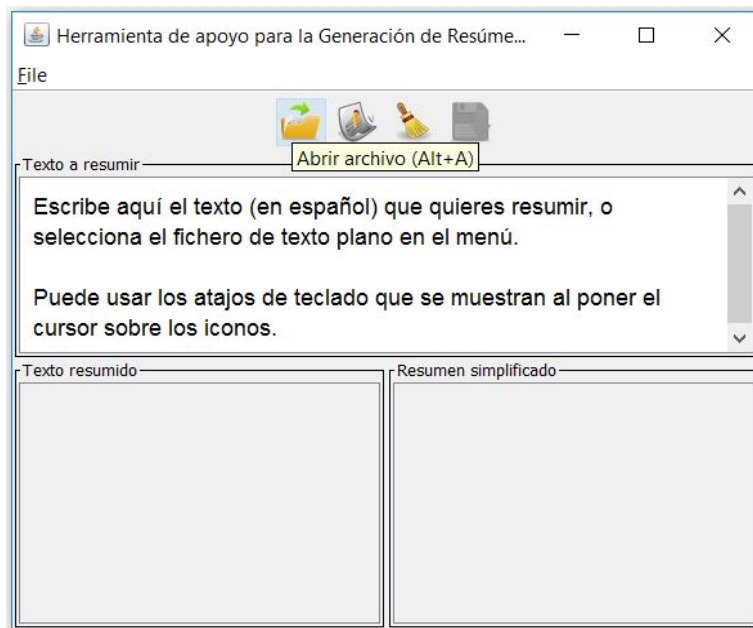


Figura 35: Cargar nuevo texto

Por último, para evitar que se cierre la aplicación por error antes de que se pueda guardar cualquiera de los resúmenes, se muestra una ventana de diálogo para confirmar que se desea salir.

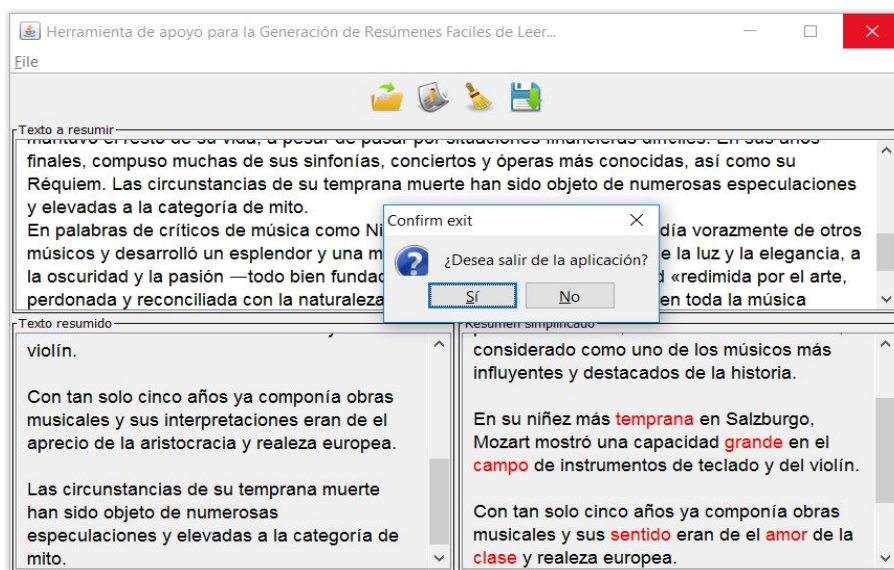


Figura 36: Cerrar aplicación

## Ap 1.2 Ejecución por consola de Eclipse

Aunque la ejecución por consola puede realizarse por una consola CMD convencional, más comúnmente conocida como *Símbolo del sistema*, para mayor comodidad los ejemplos de ejecución por consola se harán desde la herramienta Eclipse. Basta con ejecutar la clase Main.java. Más adelante se explicará cómo ejecutarlo desde el Símbolo del sistema.

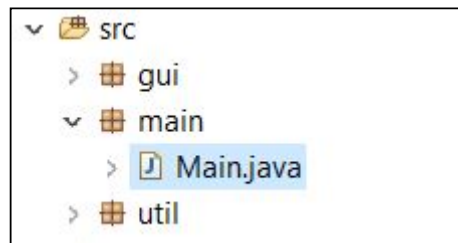


Figura 37: Ejecución por consola desde la Eclipse.

Al arrancar la ejecución por consola lo primero que se muestra es una ventana que nos permite seleccionar el fichero de texto plano cuyo contenido queremos resumir.

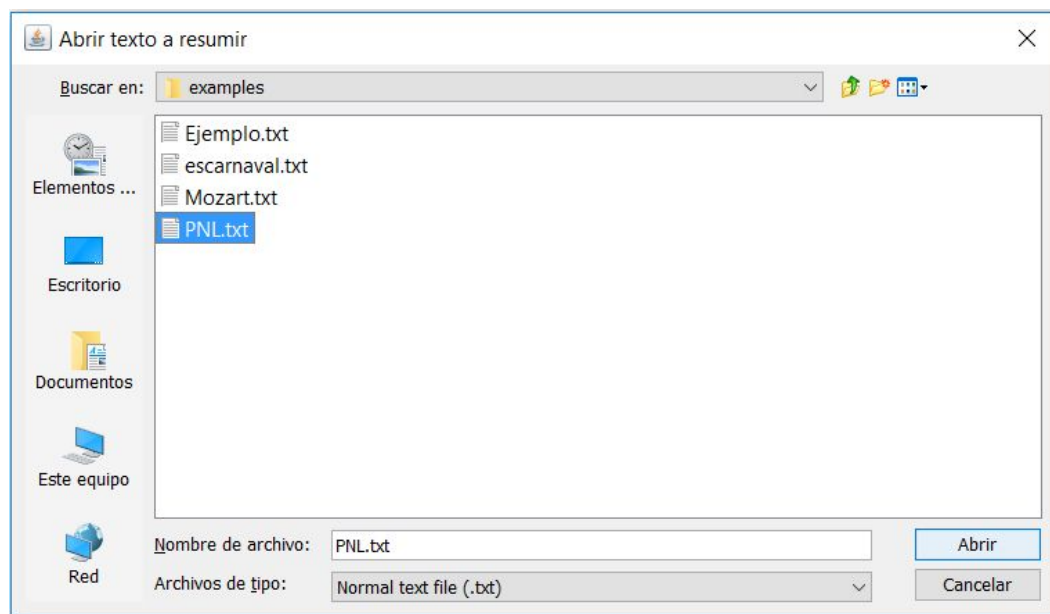
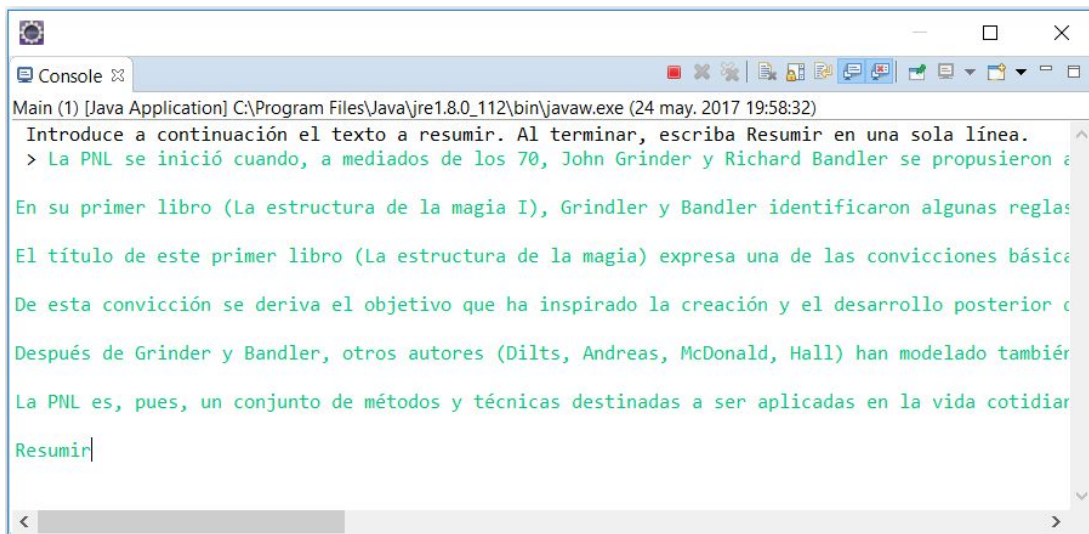


Figura 38: Cargar fichero de texto

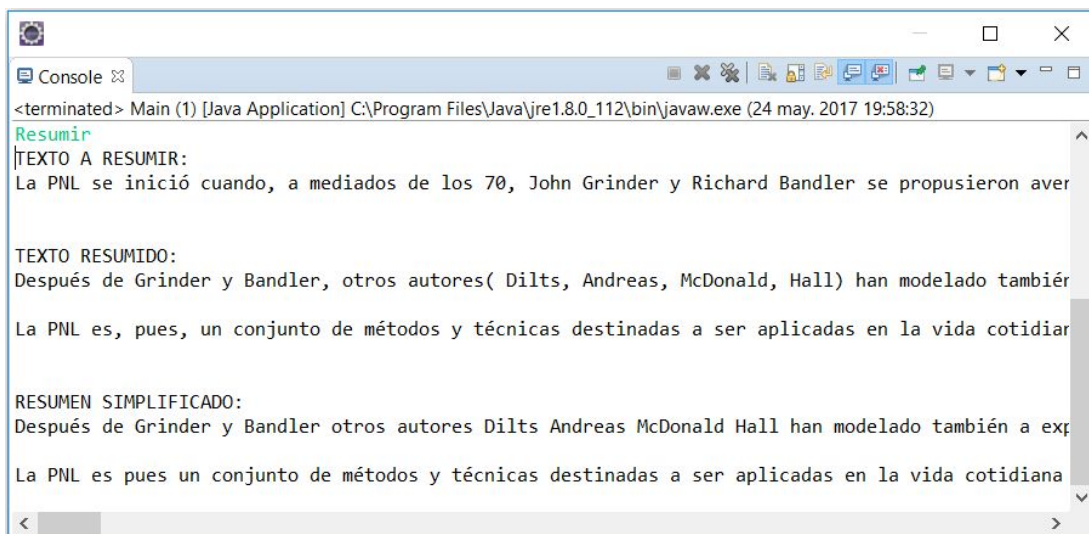
En caso de que no se disponga de un fichero de texto, cerramos la ventana y se nos indicará por consola que redactemos directamente el texto a resumir. Para terminar la redacción, basta con escribir en una línea independiente la palabra *“Resumir”*.



```
Console
Main (1) [Java Application] C:\Program Files\Java\jre1.8.0_112\bin\javaw.exe (24 may. 2017 19:58:32)
Introduce a continuación el texto a resumir. Al terminar, escriba Resumir en una sola línea.
> La PNL se inició cuando, a mediados de los 70, John Grinder y Richard Bandler se propusieron a
En su primer libro (La estructura de la magia I), Grindler y Bandler identificaron algunas reglas
El título de este primer libro (La estructura de la magia) expresa una de las convicciones básicas
De esta convicción se deriva el objetivo que ha inspirado la creación y el desarrollo posterior
Después de Grinder y Bandler, otros autores (Dilts, Andreas, McDonald, Hall) han modelado también
La PNL es, pues, un conjunto de métodos y técnicas destinadas a ser aplicadas en la vida cotidiana
Resumir|
```

Figura 39: Redacción por consola

Una vez se introduce en una línea independiente la palabra “Resumir” se comienzan a realizar una a una, las tareas encargadas de generar el texto resumido y el texto simplificado. Una vez están preparados, se muestran a continuación del texto de entrada. Tanto el texto resumido como el resumen simplificado se separan entre sí por una cabecera que indica el inicio y el comienzo de los mismos.



```
Console
<terminated> Main (1) [Java Application] C:\Program Files\Java\jre1.8.0_112\bin\javaw.exe (24 may. 2017 19:58:32)
Resumir
|TEXTO A RESUMIR:
La PNL se inició cuando, a mediados de los 70, John Grinder y Richard Bandler se propusieron aver

TEXTO RESUMIDO:
Después de Grinder y Bandler, otros autores( Dilts, Andreas, McDonald, Hall) han modelado también
La PNL es, pues, un conjunto de métodos y técnicas destinadas a ser aplicadas en la vida cotidiana

RESUMEN SIMPLIFICADO:
Después de Grinder y Bandler otros autores Dilts Andreas McDonald Hall han modelado también a exp
La PNL es pues un conjunto de métodos y técnicas destinadas a ser aplicadas en la vida cotidiana
```

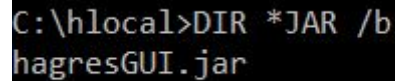
Figura 40: Salida por consola

Por último, tras generar los textos, se pregunta al usuario qué acción desea tomar (1) Guardar texto resumido, (2) Guardar resumen simplificado, (3), Guardar ambos, (0), Salir.



## Ap 1.3 Ejecución por consola de comandos

Para ejecutar el programa desde la consola de comandos basta exportar el proyecto, bien con la clase Main como principal, para trabajar solo por consola; o bien exportar con la clase MainGUI como principal, para trabajar con la interfaz gráfica. En este ejemplo exportamos el ejecutable al directorio C:\hlocal con MainGUI.java como clase principal.

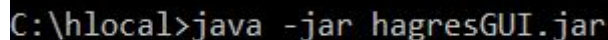


```
C:\hlocal>DIR *JAR /b
hagresGUI.jar
```

*Figura 41: Ejecutable para la ejecución por consola*

Asumimos que el usuario tiene el equipo preparado para ejecutar aplicaciones Java de modo que nos saltamos este paso. Basta entonces con ejecutar por consola la instrucción:

```
java -jar nombre_del_ejecutable.jar
```



```
C:\hlocal>java -jar hagresGUI.jar
```

*Figura 42 Ejecución por consola de comandos*

Una vez hecho esto, todos los pasos coinciden con los indicados más arriba.